VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

MODERNIZACE PORTÁLU FAKTURYONLINE.EU

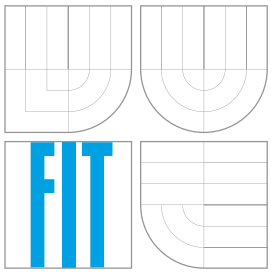
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. VOJTĚCH JAHODA

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

MODERNIZACE PORTÁLU FAKTURYONLINE.EU

UPGRADE OF FAKTURYONLINE.EU SERVICE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. VOJTĚCH JAHODA
AUTHOR

VEDOUCÍ PRÁCE — Ing. IGOR SZŐKE, Ph.D.
SUPERVISOR

BRNO 2014

Abstrakt

Cílem této práce je modernizace zavedeného webového portálu sloužícího k tvorbě a správě faktur a jeho rozšíření o nové funkce. Práce při návrhu a implementaci změn klade maximální důraz na zpětnou vazbu od uživatelů, aby pro ně byly změny opravdu přínosem a ne přítěží. Využívá širokého spektra různých technologií a při implementaci změn používá inkrementální model. Závěrem analyzuje vliv změn na návštěvnost a zjišťuje spokojenost uživatelů.

Abstract

Goal of this thesis is modernization and extension of established web portal serving to create and manage invoices. The design and implementation emphasis on feedback from users so changes are really a benefit and not a burden. Uses wide range of different technologies and implementation is done using incremental model. Analyzes the impact of changes to number of visitors and receives feedback from users.

Klíčová slova

startup, modernizace, PHP, MySQL, PDF, webový portál, tvorba faktur, Java Appler, QR kódy, SPAYD, digitální podpis PDF

Keywords

startup, modernization, PHP, MySQL, PDF, web service, invoice creating, Java Applet, QR code, SPAYD, digital signature in PDF

Citace

Vojtěch Jahoda: Modernizace portálu fakturyonline.eu, diplomová práce, Brno, FIT VUT v Brně, 2014

Modernizace portálu fakturyonline.eu

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Igora Szőkeho, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Vojtěch Jahoda
28. května 2014

Poděkování

Rád bych tímto poděkoval vedoucímu své práce panu Ing. Igoru Szőkemu, Ph.D. za jeho ochotu i odborné rady.

Obsah

1 Úvod **3**

2 Popis současného stavu **5**
2.1 Registrace . . . 5
2.2 Tvorba nové faktury . . . 5
2.3 Procházení faktur . . . 8
2.4 Adresář odběratelů a katalog zboží . . . 8
2.5 Výdajové doklady . . . 8
2.6 Statistiky . . . 9
2.7 Nastavení . . . 9
2.8 Ostatní méně důležité části . . . 9

3 Analýza spokojenosti uživatelů **11**
3.1 Průzkum . . . 11
3.2 Navržená vylepšení . . . 12
3.2.1 Vylepšení formuláře na tvorbu faktury . . . 12
3.2.2 Vylepšení práce s fakturami . . . 15
3.2.3 Ostatní změny . . . 15

4 Implementace změn **17**
4.1 Použité technologie . . . 17
4.1.1 PHP . . . 17
4.1.2 Databáze MySQL . . . 17
4.1.3 PDF . . . 18
4.1.4 FPDF . . . 21
4.1.5 JavaScript . . . 22
4.1.6 Datové formáty . . . 22
4.1.7 jQuery . . . 24
4.1.8 Java applet . . . 24
4.1.9 QR kódy . . . 25
4.2 Realizace . . . 25
4.2.1 Základ aplikace . . . 26
4.2.2 Implementace nového formuláře . . . 27
4.2.3 Přesnost aritmetiky . . . 30
4.2.4 Nastavení výpočtu DPH . . . 31
4.2.5 Číselné řady . . . 32
4.2.6 Import adresáře . . . 33
4.2.7 Grafy . . . 34

4.2.8 Párování plateb . . . 34
4.2.9 API . . . 34
4.2.10 Podepisování faktur . . . 35
4.2.11 Hromadné operace s fakturami . . . 37
4.2.12 Platební QR kódy . . . 38
4.3 Zabezpečení . . . 39
4.3.1 Důvěrnost a ochrana proti neoprávněné modifikaci . . . 39
4.3.2 Dostupnost . . . 41

5 Analýza dopadu změn **43**
5.1 Reakce uživatelů . . . 43
5.2 Vliv na návštěvnost portálu . . . 44

6 Závěr **47**
6.1 Další možnosti rozvoje . . . 47

A Obsah CD **51**

Kapitola 1

Úvod

Poslední dobou je velmi skloňovaným slovem startup. Označuje se tak nově vznikající projekt či firma s nějakou revoluční nebo alespoň inovativní myšlenkou. Nápadů je velké množství, ale těch skutečně dobrých jen velmi málo. A i takovéto projekty stárnou a jejich tvůrci získávají zkušenosti. Po několika letech je ponaučení tolik, že je potřeba provést radikální úpravy, jen aby udrželi krok s konkurencí, která začne úspěšné myšlenky okamžitě kopírovat.

V mém případě se jedná o webovou aplikaci na tvorbu faktur. Jako začínající živnostník jsem potřeboval jednoduchý nástroj na jejich tvorbu a žádný z existujících mi nevyhovoval. Vytvořil jsem ji jako absolvent gymnázia se zaměřením na programování v roce 2009. Tedy ještě před nástupem na vysokou školu. Relativně malému množství zkušeností také odpovídá kvalita zdrojových kódů a způsob řešení některých problémů.

I tak je ale aplikace velmi oblíbená. Za poslední rok ji použilo cca 5000 lidí z čehož 2000 poprvé. Celkem už uživatelé vytvořili přes 300 000 faktur. To vše bez jakékoli formy propagace nebo marketingu. Na počátku byla konkurence minimální a skládala se především z nadšenců, dnes má svůj portál většina firem nabízející účetní software. Od svého založení prošla aplikace minimálními změnami a i ty se týkaly především legislativy. Z technického hlediska byly jen opravovány chyby, zvyšováno zabezpečení a prováděny změny nutné k udržení bezproblémového chodu i s narůstajícím počtem uživatelů.

Faktura je účet za provedenou práci nebo dodané zboží či službu. Obsahuje údaje o tom, kdo ho vystavil, co je příjemci fakturováno, jakým způsobem a do kdy ji uhradit. Pojem faktura jako takový není definován v žádném právním předpisu. Přesto je pojem faktura vžitý a běžně se používá. Může (a v některých případech podle zákona musí) obsahovat i údaje o příjemci nebo o dani z přidané hodnoty.

Cílem portálu bylo primárně maximálně jednoduché generování faktur do formátu *PDF*. Původně obsahoval jen jeden formulář, až po pár týdnech přibyla možnost registrace, ukládání odběratelů, adres i faktur samotných a z toho vychází i struktura aplikace. Samotné uživatelské rozhraní tak není navrženo ideálně, protože zůstal zachován koncept jednoho formuláře. Původně se vše včetně údajů o dodavateli, který je pro jednoho uživatele ve většině případů stále stejný, muselo zadávat při tvorbě každé faktury znovu. Po přidání možnosti registrace se začaly tyto údaje ukládat a předvyplňovat, ale původní formulář stále zůstal. Vše na faktuře lze nastavit přímo z něj, od výčtu dodaného zboží či služeb po způsob výpočtu *DPH*. Údaje sice tedy není nutné zadávat pokaždé znova, ale stejně může být množství údajů pro uživatele zmatečné a v drtivé většině případů je i zbytečné.

Aplikace je napsaná v *PHP* na serverové straně, v *HTML* a *JavaScriptu* na straně klienta a data se ukládají do *MySQL*. Na serveru nebylo využito žádného frameworku, jen

pro generování PDF se používá knihovna *FPDF*. Na straně klienta je použita JavaScritpová knihovna *jQuery* a její grafické rozšíření *jQueryUI*. Web se časem přesunul na vlastní virtuální server, především kvůli SSL zabezpečení spojení. Zároveň jsem takto získal větší kontrolu nad daty i databází a bylo tak možno zavést zálohování dat i databáze v reálném čase.

Cílem práce bude modernizace tohoto webu. Změna není nikdy jednoduchá a snaha o vylepšení za každou cenu přinesla uživatelům mnoha webům více problémů než výhod, proto do ní chci maximálně zapojit uživatele a budu od nich pravidelně získávat zpětnou vazbu. Je totiž otázka, zda uživatelé očekávající časté změny už neodešli ke konkurenci a nezůstalo pouze konzervativní jádro spokojené s tím, co web v současnosti nabízí. To by mohly změny naopak odradit a je nejisté, zda by tyto ztráty příliv nových uživatelů dokázal vykompenzovat.

Kapitola 2

Popis současného stavu

Před započetím změn je nutné zrekapitulovat si, co v současnosti aplikace umí, protože z toho se bude při modernizaci vycházet. Cílem je všechny současné možnosti v maximální možné míře zachovat a jen je rozšířit. Web je rozdělen do několika částí podle jejich funkčnosti.

2.1 Registrace

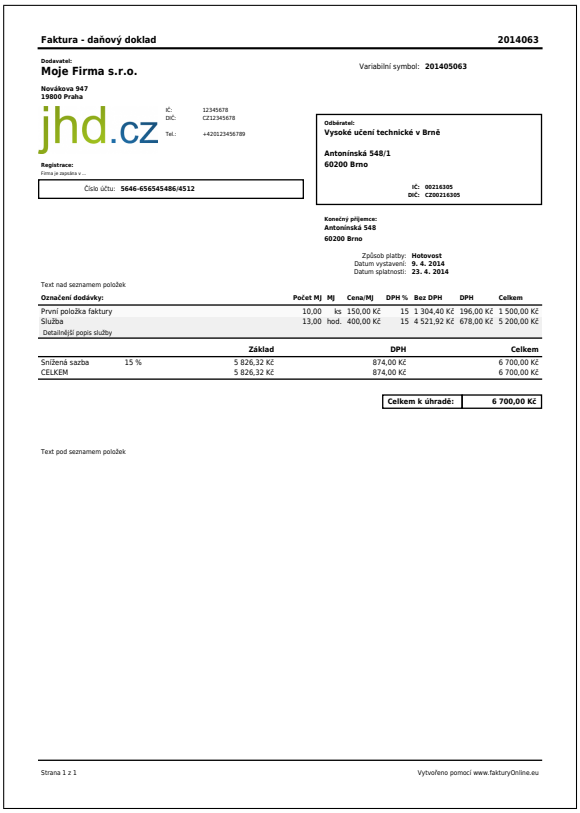
Není pro tvorbu faktury povinná, ale bez ní nelze zadané údaje o faktuře nikam uložit, jediným produktem je pak PDF dokument. Pro zjednodušení je vícekroková. V prvním kroku se zadávají pouze základní údaje, jako je *Identifikační číslo osoby* (*IČO*), e-mail a heslo. *IČO* je v České republice unikátní osmimístné identifikační číslo právnické osoby, podnikající fyzické osoby nebo organizační složky státu, proto lze očekávat, že bude jedinečné a každý vystavovatel faktur jím bude disponovat[16]. Pro přihlášení slouží právě dvojice identifikační číslo a heslo. Je neměnné a nelze registrovat subjekt se stejným číslem vícekrát, u již registrovaného čísla je nutné pokračovat obnovou hesla.

Po zadání platných údajů a odeslání formuláře je uživateli okamžitě vytvořen účet. O tom je informován e-mailem. V následujícím kroku má možnost zadat údaje o sobě, které se budou na jeho fakturách zobrazovat. Jedná se jak o údaje povinné ze zákona, jako je například *Daňové identifikační číslo* (*DIČ*), zápis firmy v obchodním rejstříku nebo její sídlo, tak o údaje dobrovolné sloužící pouze pro pohodlí odběratele, jako jsou webové stránky a e-mailový nebo telefonní kontakt. Mimo to může vložit i soubor se svým logem a razítkem, které mají na faktuře pouze estetický význam. Pokud je zadá už teď, nebude je muset zadávat při jejím vytváření. Některé z těchto údajů se systém pokusí automaticky získat z *Administrativního registru ekonomických subjektů* (*ARES*[1]) podle uživatelova identifikačního čísla, aby mu ušetřil práci. Tyto lze samozřejmě upravit a doplnit.

2.2 Tvorba nové faktury

Formulář pro tvorbu faktury samotné. Její podoba je z velké části dána zákony České republiky. Existuje několik variant podle nastavení uživatele a podle toho, zda je přihlášen,

[1] Administrativní registr ekonomických subjektů je informační systém provozovaný *Ministerstvem financí ČR*, který umožňuje vyhledávání nad ekonomickými subjekty registrovanými v České republice. Zprostředkovává zobrazení údajů vedených v jednotlivých registrech státní správy, ze kterých čerpá data (tzv. zdrojové registry). Je dostupný na http://wwwinfo.mfcr.cz/ares/ares.html.cz



Faktura - daňový doklad 2014063

Dodavatel:
Moje Firma s.r.o.

Nováková 947
19800 Praha

jhd.cz

IČ: 12345678
DIČ: CZ12345678
Tel.: +420123456789

Registrace:
Firma je zapsána v ...

Číslo účtu: 5646-656545486/4512

Variabilní symbol: 201405063

Odběratel:
Vysoké učení technické v Brně

Antonínská 548/1
60200 Brno

IČ: 00216305
DIČ: CZ00216305

Konečný příjemce:
Antonínská 548
60200 Brno

Způsob platby: Hotovost
Datum vystavení: 9. 4. 2014
Datum splatnosti: 23. 4. 2014

Text nad seznamem položek

Označení dodávky:	Počet MJ	MJ	Cena/MJ	DPH %	Bez DPH	DPH	Celkem
První položka faktury	10,00	ks	150,00 Kč	15	1 304,40 Kč	196,00 Kč	1 500,00 Kč
Služba	13,00	hod.	400,00 Kč	15	4 521,92 Kč	678,00 Kč	5 200,00 Kč
Detailnější popis služby							

		Základ	DPH	Celkem
Snížená sazba	15 %	5 826,32 Kč	874,00 Kč	6 700,00 Kč
CELKEM		5 826,32 Kč	874,00 Kč	6 700,00 Kč

Celkem k úhradě: 6 700,00 Kč

Text pod seznamem položek

Strana 1 z 1

Vytvořeno pomocí www.fakturyOnline.eu

Obrázek 2.1: Ukázka výsledné faktury

ale základní funkčnost je pro všechny shodná. Mimo vytvoření faktury je možné zobrazit si i jen náhled, jak bude vypadat. Formulář je rozdělen do logických celků a lze v něm nastavit následující:

Údaje o dodavateli

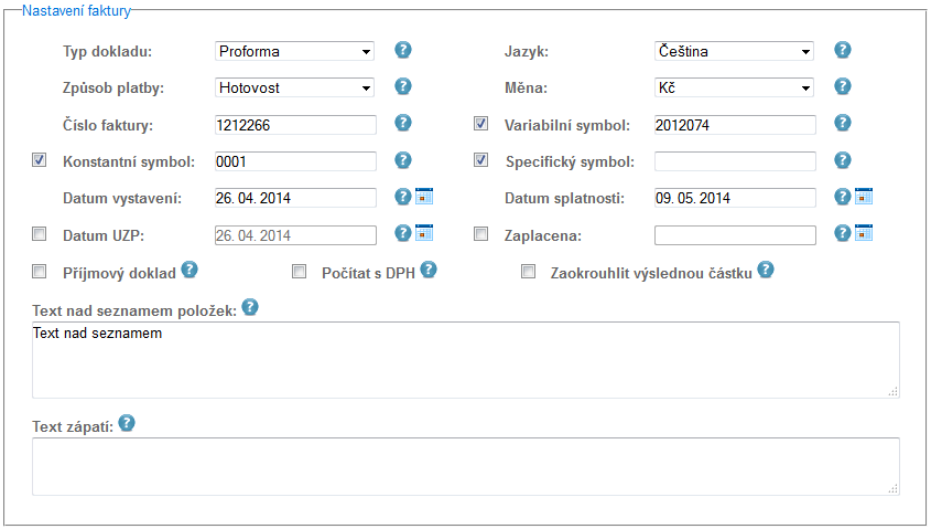
Lze zadat následující údaje: Jméno firmy, *IČO*, *DIČ*, zápis v rejstříku, číslo účtu, *IBAN*, *SWIFT*, sídlo, telefon, fax, e-mail a web. Logo a razítko se vkládají při registraci a lze je změnit v nastavení, přímo ve formuláři lze ovlivnit jen jejich viditelnost. Je možné nastavit, zda se změny mají uložit natrvalo. Standardně se tak děje.

Údaje o odběrateli

U odběratele ze zadává údajů méně. Jméno firmy, *IČO*, a sídlo. Případně je možné zadat i doručovací adresu a u plátců *DPH* i jejich *DIČ*. Při zadávání jména firmy jsou automaticky našeptávány záznamy z adresáře. Po zadání *IČO* je možné získat zbývající údaje z *ARES*. Je možné nastavit, zda se mají údaje do adresáře uložit, případně zda mají být aktualizovány. Standardně se tak opět děje.

Položky faktury

Nejdůležitější část faktury, tedy co uživatel odběrateli fakturuje, kolik toho dodal v jakých jednotkách a co za to požaduje. Při zadávání čísel se, jakmile je to možné, automaticky do-



Obrázek 2.2: Detail nastavení faktury

počítávají ostatní zatím nezadané hodnoty. Při zadávání jména se našeptávají dříve zadané položky. Počet položek na faktuře není omezen.

Při počítání s *DPH* je nutné pod seznamem položek vytvořit daňový přehled, který obsahuje celkový *základ DPH*, *DPH* a výslednou cenu pro jednotlivé sazby. Ten systém v případě potřeby sám vygeneruje.

Nastavení faktury

Tato část obsahuje všechna nastavení týkající se faktury samotné. Lze vybrat jaký typ dokladu se má vytvořit, kromě faktur umí aplikace vytvářet i cenové nabídky a *proforma faktury*. Jedná se o vžitý pojem, kterým se označuje doklad mající všechny náležitosti faktury, ale přesto se nejedná o účetní doklad. Používá se tam, kde plnění ještě neproběhlo a tak není jistá úhrada, skutečná faktura se vystaví až po jeho uhrazení. Dále je možno zvolit požadovaný jazyk faktury. Na výběr je z češtiny, angličtiny a slovenštiny.

Je zde možno i nastavit údaje potřebné k identifikaci a úhradě faktury, jako číslo faktury nebo *variabilní*, *specifický* a *konstantní symbol*. Poslední dva jsou typicky pro všechny faktury vydané jedném uživatelem, první dva systém zvládá předvyplnit jednoduše načtením čísla z faktury s nejnovějším datem vystavení (ne vytvoření) a jeho zvýšením o jedna. Novou číselnou řadu tedy lze vytvořit při vytváření nové faktury prostým zadáním jejího počátečního čísla a systém bude nadále v této řadě pokračovat. Díky této vlastnosti může být číslo faktury pouze numerické.

Lze změnit i data faktury nebo ji rovnou označit jako zaplacenou. Nad a pod seznam položek je možné vložit libovolný text, nastavit měnu faktury a zda se má počítat s *DPH*. V reakci na toto se upraví část formuláře na zadávání položek. U měny (a i měrné jednotky při zadávání zboží) je na výběr z několika nejčastěji používaných je nutno si je v nastavení aplikace povolit. Speciální funkcí je volba *Příjmový doklad*, která přidá na fakturu text potřebný k tomu, aby mohla splňovat i tuto úlohu.

Nastavení zpracování

Zde lze zvolit, jak bude s fakturou nadále zacházeno. Zda a kam se má odeslat e-mailem, zda ji ihned po vytvoření otevřít nebo například vygenerovat odkaz pro její veřejné zpřístupnění. V případě odeslání e-mailem lze nastavit její text a faktura se odešle v příloze zprávy. Výchozí text e-mailu lze změnit v nastavení a lze použít i některé proměnné, které budou nahrazeny aktuálními údaji (například dnešní datum). Zprávy se odesílají buď z uživatelem nastavené adresy nebo speciální adresy systému.

2.3 Procházení faktur

Seznam již vydaných faktur. Které sloupce se ve výpisu objeví lze ovlivnit v nastavení. Je možné omezit výpis jen na určité období podle data vystavení, splatnosti nebo uznatelného zdanitelného plnění. Stejně tak lze zobrazit pouze faktury zaplacené, čekající (nezaplacené) nebo stornované.

Obrázek 2.3: Ukázka procházení faktur

2.4 Adresář odběratelů a katalog zboží

Slouží k prohlížení již uložených záznamů, mimo to je lze i přidat, upravit a odebrat. Většinu činností lze provádět i přímo ve formuláři na vytváření faktury, jedinou skutečně unikátní je jejich mazání.

2.5 Výdajové doklady

Spíše okrajová funkčnost, o kterou uživatelé nemají příliš velký zájem. Přestože celý web je zaměřen na vytváření dokladů příjmových, tedy faktur, umožňuje evidovat a uchovávat i ty výdajové. Sekce je zcela samostatná a byla vytvořena poslední, je tedy nejmodernější. Do systému umožňuje zadat označení dokladu, potřebné časové údaje a jednotlivé částky faktury. Kvůli možným různým způsobům výpočtu DPH se automaticky nepředvyplňují další hodnoty i když by je teoreticky bylo možné dopočítat. K tomu je samozřejmě nožné přiložit neomezené množství souborů obsahují elektronickou reprezentaci dokladu, jedná se buď o *PDF* nebo *JPG*.

2.6 Statistiky

Zobrazuje základní statistiky o vydaných fakturách. Je opět možné omezit údaje jen na určité období podle data vystavení, splatnosti nebo uznatelného zdanitelného plnění. Stejně tak lze zobrazit údaje pouze pro faktury zaplacené, čekající (nezaplacené) nebo stornované. Umí spočítat:

- Počet vydaných faktur;
- Počet stornovaných faktur;
- Počet nezaplacených faktur;
- Celkové příjmy;
- Celkové výdaje;
- Výslednou bilanci.

V případě plátců DPH jsou částky rozepsány na jednotlivé sazby DPH. Pokud je použito více měn, zobrazují se pro každou měnu samostatně.

2.7 Nastavení

Je z velké části duplicitní ke schopnostem formuláře na tvorbu faktury, ale pro větší pohodlí uživatelů bylo stejně vytvořeno. Umožňuje měnit informace o firmě, které se zadávaly při registraci (vyjma identifikačního čísla), nastavení faktur a jejich zpracování.

Navíc jde zde možně změnit heslo, a adresu, ze které se odesílají e-maily. Změna není provedena ihned. Prvně je na tuto adresu zaslána zpráva, pomocí které se ověří vlastnictví adresy. Až po načtení odkazu ze zprávy je změna skutečně provedena. Dále jde nastavit výchozí splatnost faktury, četnost odesílání upomínek a zda si uživatel přeje být informován o zaslání upomínky. Při této standardně zapnuté volbě se při odeslání každé upomínky zároveň zašle e-mail i na adresu uživatele.

Mimo to lze upravit text e-mailu a upomínky. V těchto zprávách lze použít proměnné, které budou při zpracování nahrazeny skutečnými údaji. Jsou ohraničeny znakem `%` a jedná se například hodnoty `%jmenofirmy%`, `%adresa%` nebo `%splatnost%`.

Zcela specifickou částí je nastavení zobrazení. Tato část se netýká vytváření faktur, ale ovládání aplikace samotné. Lze zde ovlivnit, kolik položek na stránku se má ve výpisech zobrazit a které hodnoty se mají v seznamu faktur zobrazit. Počet najednou zobrazených sloupců není nijak omezen.

Aplikace podporuje velké množství měn a jednotek, proto je zde možnost si vybrat, které se mají při vytváření faktury nabídnout. Při současném zobrazení i těch nepotřebných by bylo komplikované najít tu požadovanou.

2.8 Ostatní méně důležité části

Přehled

Uvítací stránka uživatele po jeho přihlášení. V případě potřeby se zde zobrazují důležitá upozornění či varování. Například o plánovaných odstávkách.

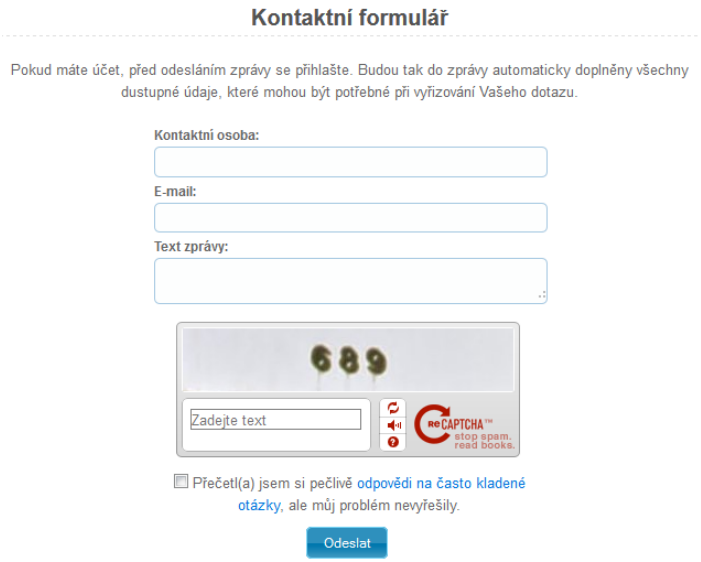
Standardně zobrazuje stručné informace o tom kolik faktur, druhů zboží, odběratelů a výdajových dokladů má uživatel v systému uložených. Taktéž uživateli zobrazuje, jak velký prostor zabírají jeho soubory s výdajovými doklady.

Často kladené otázky

Seznam často kladených otázek společně s detailními odpověďmi na ně. Jednoduchá, ale důležitá součást webu, která umožnila snížit původní množství dotazů na jeho zlomek.

Kontakt

Jediný způsob, jak kontaktovat autora aplikace je kontaktní formulář. Ten je přístupný přihlášeným i nepřihlášeným uživatelům, jen v případě těch druhých je kvůli spamu požadováno opsání kódu z kontrolního obrázku. Před odesláním zprávy musí uživatel odsouhlasit, že si přečetl často kladené otázky. Zpráva je odesílána standardně na e-mail správce a systém do ní doplní údaje užitečné pro rychlé vyřízení zprávy. Například *IČO*, *ID* uživatele, jeho *IP*, použitý prohlížeč a stav *SSL*.



Obrázek 2.4: Ukázka kontaktního formuláře

Kapitola 3

Analýza spokojenosti uživatelů

Před samotným plánováním změn je potřeba zjistit, co od služby uživatelé vlastně očekávají. Tento projekt má výhodu, že nemusí složitě odhadovat, co od něj uživatelé chtějí, ale může využít svou uživatelskou základnu a zeptat se jich, co jim na portálu vyhovuje a co by naopak změnili nebo přidali.

3.1 Průzkum

Bylo tedy potřeba vytvořit dotazník, do kterého by tyto informace mohli vložit. Měl za cíl zjistit, nakolik jsou uživatelé spokojeni s jednotlivými částmi portálu a na co se tedy zaměřit. K tomu posloužily známkovací otázky. Důležité bylo také zjistit, proč případně jsou či nejsou spokojeni. Zároveň jej bylo využito k získání testovací skupiny, která bude mít přednostně přístup k novinkám. Časté změny související s nasazením změn by asi většinu uživatelů spíše odradily, než potěšily. Pro budoucí použití byly vloženy i otázky o možnosti zpoplatnění.

Otázka	**Způsob odpovědi**
Jak jste spokojeni s aplikací jako celkem?	Stupnice 1-5
Jak jste spokojeni s těmito součástmi?	
Formulář na tvorbu faktury	Stupnice 1-5
Statistiky	Stupnice 1-5
Správa výdajů	Stupnice 1-5
Adresář	Stupnice 1-5
Zboží a služby	Stupnice 1-5
Co se Vám na aplikaci líbí?	Textové pole
Co se Vám na aplikaci nelíbí?	Textové pole
Měli byste zájem přednostně testovat nové funkce?	Ano / Ne
Přáli byste si přidat nějaké funkce?	Textové pole
Byli byste ochotni za další funkce platit?	Ano / Ne
Jakou částku byste byli ochotni platit za rok?	Číslo

Tabulka 3.1: Seznam otázek v dotazníku

Známkování na stupnici 1-5 probíhalo jako ve škole, tedy 1 byla nejlepší možnou známkou. Uživatelé měli také možnost své výhrady a návrhy na vylepšení sdělit slovně. Veškeré údaje se ukládaly přímo do databáze. Při tvorbě dotazníku byl kladen důraz na jeho struč-

nost, žádná odpověď nebyla povinná, aby tak nějakou odpověď poskytlo co nejvíce uživatelů. Průzkumu se zúčastnilo 468 registrovaných uživatelů. Získané známky byly zprůměrovány a výsledky vyhodnoceny.

Celý portál obdržel známku 1,68, z čehož lze usuzovat, že většina uživatelů je s ním celkově spokojena. Z jednotlivých sekcí dopadly nejhůře statistiky s 3,41. Naopak nejlépe si vedla správa výdajů s 1,26, za nimi následoval adresář s 2,44 a katalog zboží a služeb s 2,75. Samotný formulář pro tvorbu faktury dopadl s 2,89 obstojně. Uživatelé mu dle očekávání nejčastěji vytýkali jeho složitost a nepřehlednost. Mimo to bylo získáno velké množství požadavků na různá vylepšení. Ta smysluplná, realizovatelná a opakující se byla postoupena k dalšímu hodnocení. Podařilo se také úspěšně získat skupinu 73 budoucích testerů, což je velmi důležité pro testování novinek.

Také byl vytvořen nástroj, který ukládá informace o používání formuláře. Ten sledoval, jak dlouho uživatelům vytvoření faktury průměrně trvá. Takto bude možné při vyhodnocování porovnat subjektivní názor uživatele na používání nového formuláře s objektivně naměřenými hodnotami.

3.2 Navržená vylepšení

Tato část práce stručně popisuje zvažovaná vylepšení, navržená na základě informací získaných z průzkumu, u nichž byla ověřena proveditelnost. Tento seznam byl předložen uživatelům a byla od nich získávána zpětná vazba. U každého návrhu mohli hodnotit, zda je pro ně bude změna přínosná nebo naopak nějakým způsobem omezující. V případě, že na ně nebude mít žádný vliv, hodnotit ji nemuseli. Průzkumu se zúčastnilo 189 uživatelů. Podle něj se potom určilo, zda a kdy budou změny implementovány.

3.2.1 Vylepšení formuláře na tvorbu faktury

Změny byly rozděleny do tří částí, první skupina obsahuje ty, které se přímo týkají formuláře na tvorbu faktury.

Redesign formuláře na tvorbu faktury

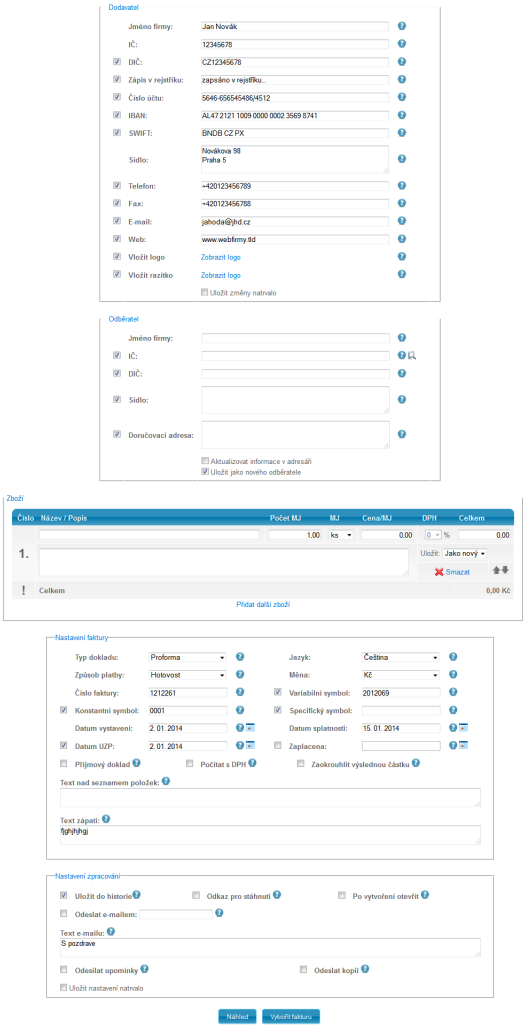
Při redesignu (viz obrázek 3.2) jsem se tedy zaměřil především na zvýšení jeho přehlednosti. Zároveň jsem ale nechtěl uživatele připravit o žádnou z již dříve existujících možností, takže koncept jednoho formuláře zůstal zachován, jen byl změněn jeho vzhled a způsob zobrazení. Většina položek je na první pohled skrytých a dají se zobrazit jen pomocí *JavaScriptu*.

Návrh nového formuláře je o poznání jednodušší. Přímo na stránce zůstaly jen pole pro našeptávače odběratele a zboží a některá základní nastavení. Všechny ostatní části byly byly skryty a lze je vyvolat kliknutím na patřičné tlačítko, typicky vedle vstupního pole.

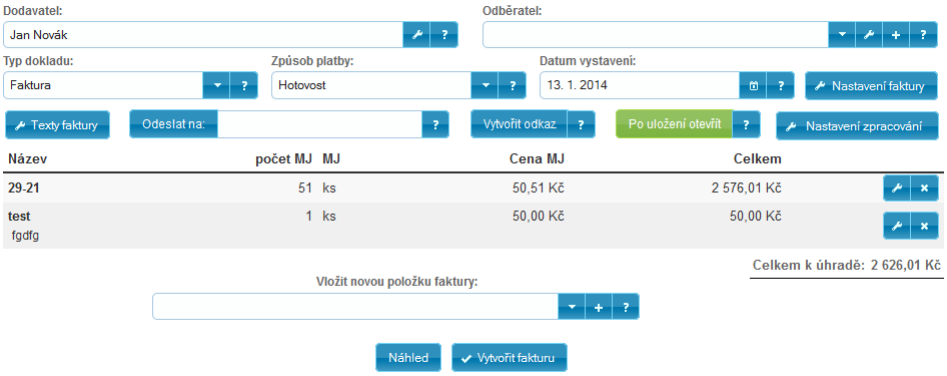
Ve výsledku je tak pro jednotlivá vstupní pole více místa, vše přesto zůstalo zachováno a formulář působí značně přehlednějším dojmem. Členění na části zůstalo zachováno stejné, jaké bylo v původním formuláři (viz obrázek 3.1).

Vyšší přesnost aritmetiky

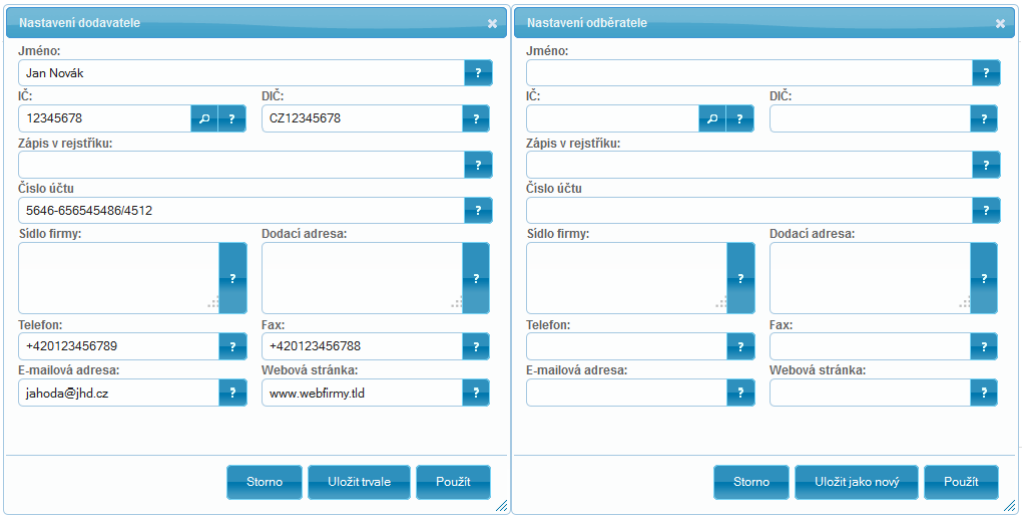
Díky omezením daným použití aritmetiky v plovoucí řádové čárce někdy dochází k nepřesnostem u výpočtů, především při zaokrouhlování. Občas se stává, že číslo s desetinnou částí 0,5 je ve skutečnosti uloženo jako 0,49 a jeho zaokrouhlení chybně proběhne směrem



Obrázek 3.1: Původní formulář pro tvorbu faktury byl velmi rozsáhlý a nepřehledný



Obrázek 3.2: Návrh nového formuláře



Obrázek 3.3: Méně podstatné prvky byly skryty do vyskakovacích oken

dolů[5]. I když se problém zdá marginální a vyskytuje v minimu případů, takovéto chyby na fakturách nejsou pro uživatele nic příjemného.

Výpočty DPH a zaokrouhlování

Podle *Zákona o dani z přidané hodnoty* existuje několik způsobů, jak postupovat při výpočtu *DPH*[1] a jejich kombinacemi vzniká značné množství potenciálních scénářů. Kromě možnosti *DPH* k ceně přičíst nebo ji vypočítat z výsledné ceny ji lze i několika způsoby zaokrouhlovat. Různí účetní mají na tuto problematiku různé názory. Navíc pro každého může být výhodnější jiná varianta. Ale bylo by možné dát uživateli možnost, ať si výpočet nastaví podle sebe. Aby se takto složité výpočty nemusely počítat opakovaně, bylo by dobré do databáze ukládat všechny vypočtené hodnoty.

QR platební kódy

Podporují je některé banky a mohou obsahovat všechny údaje potřebné k uhrazení faktury[24]. Uživatel pak nemusí nic vyplňovat, pouze vhodným zařízením načte QR kód a provede samotný převod.

Velikost písma

Současný výstup z PDF se soustředí především na to, aby se na fakturu bezpečně vešlo vše podstatné. Nicméně písmo může být po vytisknutí opravdu hůře čitelné. Proto by mohlo být vytvořeno více šablon faktur a uživatel by si mohl zvolit si i nějakou s většími omezeními na délky textu a větší velikostí písma.

Párování plateb s výpisy z banky

Jedná se o prosté hromadné automatické označení faktur jako zaplacených podle výpisu z banky. Snad každá banka umožňuje v elektronickém bankovnictví provést export dat do

[1] Příslušný zákon online na http://business.center.cz/business/pravo/zakony/dph/cast1h2d6.aspx

nějakého strojově čitelného formátu. Aplikace ho projde, porovná variabilní symbol a částku s nezaplacenými fakturami a pokud najde shodu, označí fakturu jako zaplacenou.

Znaky v číslech faktur

Jakkoli to zní nelogicky, nemalá část firem používá více číselných řad a potřebuje je od sebe nějakým způsobem jednoduše, ale zároveň jednoznačně odlišit. A k tomuto účelu je vhodný právě abecední znak na začátku či konci čísla. Ale současný způsob získání následujícího čísla to neumožňuje.

3.2.2 Vylepšení práce s fakturami

Zde jsou shrnuta vylepšení, která jsou přímo spojena se správou faktur.

Rozhraní pro strojovou komunikaci s portálem (API)

Službu používá velké množství různých e-shopů, kterým by vyhovovalo kdyby do něj nemusely vkládat údaje pro tvorbu faktur ručně, ale bylo možné jejich vkládání automatizovat.

Export faktur

Formát ukládání adresy není kompatibilní se žádným standardizovaným formátem. Je to dáno tím, že původně byl program určen pouze ke generování PDF a nebyl žádný důvod omezovat uživatelům formátování adresy. Touto vlastností si získal velké množství zahraničních klientů, kteří potřebují vytvářet faktury v češtině, ale omezení běžných programů toto nedovolí. Proto je celá adresa uložena v jednom řetězci. Pro většinu ostatních formátů by bylo nutné jednotlivé položky rozdělit. To by navíc obnášelo nutnost aktualizace adresáře a i při stahování dat z *ARES* by to pro větší část uživatelů byla zbytečná komplikace. Ale bylo by možné exportovat více faktur do jednoho PDF nebo archivu.

Hromadné operace

Aktuálně je nutné všechny operace s fakturami provádět pro každou fakturu jednotlivě, přestože některé z nich by bylo možné bez větších problémů provádět i hromadně. Například by bylo reálné více faktur najednou označit jako zaplacené, odstranit nebo smazat.

Vyhledávání faktur

V současné verzi lze faktury filtrovat pouze podle dat a jejich stavu. V nové verzi by bylo možné faktury filtrovat například i podle jména firmy odběratele.

Statistiky

Nejhůře hodnocená část původního webu je v současnosti opravu minimalistická a až přespříliš jednoduchá. Nová verze by mohla by umožnit zobrazit statistiky zobrazit i podle jednotlivých odběratelů a porovnat jejich důležitost nebo vytvořit jejich grafickou reprezentaci.

3.2.3 Ostatní změny

Vylepšení týkající se ostatních částí portálu.

Import adresáře

Zde neexistuje žádný masově používaný standardizovaný formát. Proto bude umožněn import z *CSV*. Jen bude potřeba zvolit vhodný formát, aby do něj mohla exportovat většina uživatelů.

Přihlašování

Možnost mít více uživatelských účtů k jednomu IČO a naopak stejné účet pro více IČO. Původní úvaha vycházela z toho, že každý ekonomický subjekt musí mít státem přidělený identifikátor, proto ho lze použít pro přihlášení. To se ukázalo jako nepravdivé, protože faktury mohou vystavovat i ekonomické subjekty bez IČO, například při pronajímáni nemovitosti nebo provozování umělecké činnosti. Navíc u větších firem, kde se o faktury stará více lidí, není vhodné, pokud stejné údaje používá více uživatelů. A naopak je zbytečné, když se uživatel musí pro práci s jiným IČ přihlašovat a odhlašovat.

Přehlednost nastavení

Původní nastavení bylo postupně rozšiřováno o další a další možnosti, které v současnosti nejsou ani příliš dobře logicky rozděleny. V závislosti na podobě a možnostech nové verze aplikace by bylo vhodné členění nastavení kompletně přepracovat.

Kapitola 4

Implementace změn

Abych se vyhnul riziku, že vytvořím produkt, který uživatelé nechtějí, budou změny probíhat postupně po částech v iteracích. Pořadí bude stanoveno podle uživatelského hodnocení jejich přínosu. Budou nabízeny vybraným dobrovolníkům k odzkoušení a ke každé bude získávána zpětná vazba. Je proto nutné udržet kompatibilitu se současným systémem a podobou databáze. Ani jedna z těchto podmínek by neměla být omezující, naopak by se měl snížit počet možných chyb díky tomu, že některé odzkoušené části kódu bude možné zachovat.

4.1 Použité technologie

Tato část práce popisuje jednotlivé technologie, které byly zvoleny pro různé části implementace a jejichž znalost bude při jejím popisu očekávána.

4.1.1 PHP

*PHP: Hypertext Preprocessor*je skriptovací jazyk určený primárně pro vytváření dynamických webových stránek[6]. Je nutné ho kombinovat s nějakým dostupným webovým serverem, například *Apache* nebo *nginx*. Díky jeho rozšířenosti existuje velká komunita a široký výběr modulů doplňujících jeho základní funkcionalitu.

Jak již je v textu zmíněno, i pro implementaci nové verze použiji čisté *PHP*. Prvním důvodem je potřeba provádět úpravy po jednotlivých iteracích a tedy zachování co největšího propojení s již existujícími součástmi webu. Ale i nebýt tohoto zvolil bych si tento jazyk, protože nabízí dobrý poměr použitelnosti a rychlosti. V poslední době se objevuje velké množství frameworků, které slouží především pro zpřehlednění kódu a zjednodušení práce programátora, ale už ze své podstaty mají větší nároky na zdroje než *PHP* samotné. K režii interpretu totiž nevyhnutelně přibude režie frameworku.

4.1.2 Databáze MySQL

Databázový systém dostupný pod otevřenou licencí[1]. Oproti některým ostatním *SQL* serverům má jistá omezení, protože byl od začátku zaměřen především na rychlost. Nicméně žádná z nich nejsou natolik zásadní, aby převážily nad jeho výhodami. Od převzetí vývoje

[1] http://www.mysql.com/

společností *Oracle* se začala více prosazovat nástupnická větev pod názvem *MariaDB*[2]. Jejím hlavním vývojářem je původní zakladatel *MySQL*. Disponuje některými výhodami a podle některých testů je výkonnější. Nicméně *MySQL* je dlouhodobě odzkoušené a *MariaDB* je plně kompatibilní, proto není problém na ni kdykoli v budoucnu přejít.

MySQL nabízí několik databázových úložišť. Bylo zvoleno *MyISAM*, které se na serveru ukázalo jako mnohem rychlejší. InnoDB disponuje některými pokročilejšími funkcemi, jako například zamykání jednotlivých řádků nebo existující správou relací, ale to se pravděpodobně negativně projevilo na jeho výkonu.

Pro komunikaci s databází bude použit *PHP* modul `php_mysql`. I když je již v současnosti označen organizací *PHP Group* jako zastaralý[18], ani tato informace nezmiňuje důvod. Neobsahuje žádné chyby nebo bezpečnostní rizika mimo absenci některých novějších funkcí. Způsoby přístupu bohužel nelze kombinovat, takže zbývá buď možnost celý web převést na doporučovanou knihovnu nebo používat rozhraní označení jako zastaralé i v novém kódu. Nějaký ekvivalent všech konstrukcí realizovaných pomocí tohoto rozšíření zvládá i `php_mysqli` a není tak problém provést nahrazení. Bohužel jejich rozhraní si nejsou natolik podobná, aby bylo možné vytvořit spolehlivý skript na převod.

4.1.3 PDF

Portable Document Format vytvořený firmou *Adobe* v roce 1993 slouží k ukládání textových dokumentů odvozený od jazyka *PostScript*. Dokáže vždy přesně zachovat vzhled dokumentu bez ohledu na hardware, software či rozlišení, ve kterém byl vytvořen nebo je zobrazován. Přímo v něm jsou uloženy i fonty potřebné k zobrazení, umožňuje vkládání obrázků a od novějších verzí i digitální podpis dokumentu či jeho části, vytváření interaktivních formulářů nebo spouštění jednoduchých skriptů v jazyce JavaScript[1]. Společně s velkým rozšířením nástrojů sloužícím k jeho zobrazení (především Adobe Reader) se jedná o ideální formát, pokud chceme zaručit přesné zobrazení dokumentu pro co nejvíce uživatelů.

Při navrhování formátu nebyl kladen velký důraz na přehlednost zdrojového kódu dokumentu. Při otevření libovolného PDF dokumentu v textovém editoru můžeme dokonce nabýt dojmu, že se jedná o soubor binární. Tak tomu ale není a formát je ve skutečnosti strojově dobře zpracovatelný a dokonce i editovatelný. 1. června 2008 ho společnost Adobe otevřela veřejnosti a byl publikován organizací *International Organization for Standardization* jako *ISO/IEC 32000-1:2008*[1]. Tento dokument má 756 stran a je v něm detailně, ale zároveň srozumitelně, popsán.

Datové typy

PDF zahrnuje 8 základních datových typů: logický, celá čísla, čísla v plovoucí čárce, řetězce, identifikátory, pole, slovníky, proudy a prázdný objekt.

Boolean Logický typ reprezentující true nebo false.

Integer Celočíselný typ skládající se z jedné či více číslic, který může být uvozen znaménkem. Jeho typický rozsah je -2 147 483 648 až 2 147 483 647, ale může se lišit podle konkrétní architektury.

[2] https://mariadb.org/

Real Zapsán stejně jako `Integer`, navíc může obsahovat tečkou oddělenou desetinou část. V případě, že je v nějakém místě dokumentu očekávána reálná hodnota, ale toto číslo nedisponuje desetinnou částí a mohlo by tak být považováno za celočíselné, bude automaticky přetypováno.

String Řetězce se skládají z tisknutelných ASCII znaků, které jsou uzavřeny v jednoduchých závorkách. Pokud je potřebujeme použít, musíme tak učinit v páru, tj. použít obě dvě. Pokud tato podmínka není splněna, je třeba před ně připojit zpětné lomítko. Řetězce je také možno zapsat v šestnáctkové soustavě, v tomto případě budou ohraničeny v ostrých závorkách. Tento zápis je vhodný pro kratší sekvence binárních dat.

Name Objects Identifikátory, mohou se skládat z libovolných ASCII znaků mimo nulového. Jsou uvozeny znakem `/`, za `#` je očekáván hexadecimální zápis znaku. Tím je potřeba zapisovat speciální a netisknutelné znaky.

Array Objects V PDF jsou pole heterogenní, uzavřeny v hranatých závorkách, jednotlivé hodnoty jsou odděleny mezerami. Mohou obsahovat kterékoli objekty včetně polí samotných.

Dictionary Objects Jedná se o asociativní tabulku obsahující dvojice klíč — hodnota. Klíč musí být typu `Name`, hodnota může být kteréhokoli typu včetně slovníku samotného. Jsou z každé strany ohraničeny dvojicí ostrých závorek `<< >>`.

Stream Objects Sekvence binárních dat. Používají se pro části souboru, kde by běžný string už nebyl vhodný, například obrázky. Zápis je následující

```
<</Length 299
/Filter /FlateDecode
>>
stream
...binární data dlouhá 299 bytů...
endstream
```

Jak je vidět, samotná data jsou ohraničena klíčovými slovy `stream` a `endstream` a předchází jim slovník. Ten může mít různé parametry určující obsah, podstatný je `/Length`, který je povinný a určuje délku streamu. Pokud na něj byl aplikován filtr, počítá se s jeho výsledkem. `/Filter` určuje, co se má s daty před použitím provést, mohou být například komprimována. Je možné aplikovat více filtrů zároveň, pak musí být zapsány ve stejném pořadí, v jakém mají být provedeny.

Null Object Neexistující objekt, klíčové slovo `null`. Pokud se na ni nastaví například nastaví hodnota klíče ve slovníku, výsledek je stejný, jako by byl klíč zcela vynechán.

Indirect Objects Nepřímé objekty. Jedná se o speciální typ, protože každý objekt v PDF může být nepřímý. Například i náš předchozí *stream*.

```
10 0 obj
<</Length 299
/Filter /FlateDecode
>>
stream
...binární data dlouhá 299 bytů...
```

```
endstream
endobj
```

Tímto získal identifikátor 10. 0 je regenerační číslo, které určuje počet znovupoužití identifikátoru. Objekt samotný je uzavřen mezi klíčová slova `obj` a `endobj`. Pokud ho budeme chtít někde vyvolat, uděláme to s použitím identifikátoru, generačního čísla a znaku `R` následovně:

```
<< /Stream 10 0 R >>
```

Takto jsme vytvořili slovník, kde klíč `Stream` odkazuje na náš původní *stream*.

Struktura dokumentu

Dokument se skládá ze 4 základních částí:

Hlavička Pouze první řádek souboru, který specifikuje verzi dokumentu ve tvaru:

```
%PDF-1.3
```

Aktuálně existující verze jsou 1.0 až 1.7.

Tělo Obsah samotného dokumentu, který bude zobrazen uživateli. Zde se nachází texty, obrázky, multimédia, fonty atd.

xref tabulka Tabulka křížových referencí, která umožňuje náhodný přístup k jednotlivým objektům v souboru a není tedy potřeba číst ho celý sekvenčně. Může vypadat například takto:

```
xref
0 6
0000000003 65535 f
0000000017 00000 n
0000000081 00000 n
0000000000 00007 f
0000000331 00000 n
0000000409 00000 n
```

`xref` uvozuje začátek tabulky, dvojice čísel odděluje sekci tabulky a nastavuje její číslování. V tomto případě bude tabulka obsahovat 6 záznamů číslovaných od nuly. Samotné reference mají vždy délku 10 bytů. Prvních deset číslic určuje offset objektu od začátku dokumentu, pokud je kratší, doplní se zleva nulami. Mezerou je odděleno *generation number*, které se používá pouze při mazání a znovupoužívání objektů. Další mezerou je oddělen příznak určující, zda je objekt použit. `f` znamená, že je volný, `n` použitý. Následuje dvojice znaků pro ukončení řádku. U volných objektů desetimístné číslo neobsahuje pozici objektu, ale číslo dalšího volného. Na první smazaný objekt by nemělo co odkazovat, proto je první záznam v tabulce vždy speciální volný a má „generation number“ `65535`. To se při smazání i znovu použití se navýší vždy o 1. Takto vytvořený řetěz umožňuje volná čísla jednoduše použít znovu. Při úpravách dokumentu může mít tabulka více sekcí a dokonce může existovat více tabulek.

Trailer Obsahuje informaci o pozici poslední `xref tabulky` a některé důležité informace o identitě dokumentu. Povinné jsou parametry `/Size` (udává počet záznamů v xref tabulce) a `/Root` (nepřímá reference na objekt obsahující katalog) Fakt, že tyto údaje jsou na konci dokumentu znamená, že je vhodné jej začít číst od konce, což není ideální při jeho načítání z webu. Ukončen je sekvencí `%%EOF`, značící konec dokumentu.

Struktura těla

Uspořádání objektů v těle dokumentu je hierarchické. Jeho kořenem je *Document Catalog dictionary*, jehož umístění je specifikováno v *Traileru*.

Document Catalog dictionary Jak název napovídá, jedná se o slovník. Má dva povinné parametry — `Type`, který specifikuje, že se jedná o `Catalog` a `Pages`, ten obsahuje nepřímou referenci na další objekt, který je kořenem stromu stránek. *Document Catalog* také může obsahovat informace o použité verzi PDF, rozložení stránek, výchozím přiblížení atd. nebo odkazy na objekty obsahující přehled obsahu, licenční ujednání, informace o autorovi apod.

Page Tree Pomocí stromu stránek se přistupuje k jednotlivým stranám dokumentu. Stromové uspořádání bylo zvoleno kvůli úspoře paměti a možnosti rychle otevřít velké dokumenty. Obsahuje dva typy objektů, uzly a listy. Uzly mohou odkazovat na další uzly nebo rovnou strany. V obou případech se jedná o slovníky. Teoreticky by bylo možné na všechny strany odkázat hned v první uzlu, to by ale negativně ovlivnilo rychlost práce s dokumentem.

Page Tree Nodes Každý uzel obsahuje povinně informaci o tom, kdo je jeho rodič (mimo kořene), na kolik stran on a jeho potomci odkazují a pole potomků.

Page Objects Strana samotná, objekt může mít mimo obsahu samotného velké množství parametrů. Kdo je jeho rodičem, kdy byl naposledy modifikován, jaké potřebuje zdroje, jaký má formát, rotaci, ořezání, odsazení, zmenšený náhled a mnoho dalších.

4.1.4 FPDF

PHP třída umožňující vytváření PDF dokumentů pouze pomocí prostředků *PHP*[3]. Zvládá všechny základní operace s *PDF*. Vkládání textů do bloků, různé fonty, vkládání obrázků v několika formátech, kreslení základních grafických objektů atd. Při použití knihovny se musí detailně specifikovat, co se má kam vykreslit jakým způsobem. Pracuje s kontextem, takže se prvně nastaví jakým fontem psát, jakou má mít velikost a barvu a kde se má zobrazit a až poté se použije funkce na jeho výpis. Naštěstí při výpisu textových buněk dochází k automatickému posunu kurzoru, takže s trochou praxe není vytvoření požadovaného výstupu větším problémem.

Umožňuje předefinovat funkci vypisující záhlaví a zápatí s jednoduchým vložením čísla stránky a celkového počtu stran, což je právě při vytváření faktur velmi užitečné, protože společně s automatickým posunem kurzoru tím vyřeší veškeré problémy se stránkováním za nás a umožní spolehlivě vytvářet prakticky neomezeně dlouhé faktury.

Byla ale vytvořena v roce 2002, v poslední době už se příliš nevyvíjí a některé podstatné věci jí chybí. Byla vydána pod svobodnou licencí, takže se brzo objevilo několik nástupců.

tFPDF

Jedná se o třídu *FDPF*[4] rozšířenou o podporu *UTF-8* se schopností vložit do dokumentu pouze opravdu použité znaky z fontu. Při použití *FPDF* v české lokalizaci je nutné do výstupního souboru vložit celý font. To znamená v případě využití standardního i tučného řezu písma velikost výsledného souboru i přes 500kB. *tFPDF* je schopen z fontu vybrat

[3] http://www.fpdf.org/

[4] http://www.fpdf.org/en/script/script92.php

pouze ty znaky, které jsou k jeho zobrazení skutečně potřeba a tím srazit velikost soubor i pod 50kB.

TCPDF

Další třída pro práci s *PDF*[5]. Je postavena na základu *FPDF*, kterou rozšiřuje o velké množství možností, umožňuje například automatický převod z *HTML*, podepisování dokumentů digitálním podpisem, generování čárových kódů a mnoho dalšího.

FPDI

Rozšiřuje *FPDF* i *TCPDF* o možnost importu již existujícího dokumentu[6], ten pak lze jednoduše upravovat a není potřeba ho celý generovat znovu. Je velmi užitečná například pro hromadné exporty nebo dodatečné podepisování.

4.1.5 JavaScript

Je multiplatformní prototypově orientovaný jazyk, který se nejčastěji používá ve webových prohlížečích. I přes svůj název nemá s *Javou* prakticky nic společného, pojmenování byl jen marketingový tah, předtím vystřídal několik dalších označení[7]. Existuje mnoho jeho implementací, dokonce i v každém z prohlížečů se liší. Protože nástroje na manipulaci s objektovým modelem stránky nebyly součástí původního JavaScriptu a nebyl tedy nikdo, kdo by je standardizoval, ujala se této role organizace W3C[20], která se stará i o vytváření HTML standardů. V poslední době díky tomu došlo ke značnému zlepšení a nové schopnosti jsou do prohlížečů zaváděny ve stejné podobě, což značně zlepšuje prostředí pro vývojáře a umožňuje jim rychleji adaptovat nové technologie.

4.1.6 Datové formáty

Aplikace díky komunikaci s různými službami a možnostmi exportu a importu využívá relativně velké množství textových formátů pro uložení a přenos dat.

JavaScript Object Notation (JSON)

Je textový standard pro přenos dat. Má dva základní datové typy, pole a objekt[3]. Pole může být heterogenní a není asociativní. Data v objektech jsou ukládána vždy ve dvojicích klíč – hodnota, přičemž i ta může být dalším objektem. Jeho výhodou oproti například XML je větší přehlednost (samozřejmě při vhodném formátování), menší datová náročnost a jednodušší zpracovatelnost, přičemž pro většinu úkonů je bez problémů dostačující. Mezi nevýhody patří absence možnosti nastavit kódování, což ale při přenosu přes *HTTP* lze jednoduše vyřešit hlavičkou `Content-Type`. Pro netisknutelné znaky používá sekvence uvozené zpětným lomítkem, například \n nebo \t, které jsou známé i z ostatních programovacích jazyků. Ostatní netisknutelné znaky jsou uloženy jako `\uXXXX`, přičemž `XXXX` reprezentuje hexadecimální číslo znaku v *UTF-16*. Ač tedy *JSON* zvládá přenášet binární data, není k tomu příliš vhodný, protože 1 netisknutelný znak zabere 6 bytů. Je vhodnější data prvně

[5] http://www.tcpdf.org/

[6] http://www.setasign.com/products/fpdi/about/

[7] Rozhovor s autorem na toto téma dostupný na http://www.infoworld.com/d/developer-world/javascript-creator-ponders-past-future-704

zakódovat pomocí *Base64* (viz odstavec 4.1.6), tak získáme tisknutelnou reprezentaci dat, která není zdaleka tak náročná.

```
{
  "jmeno": "Jan",
  "prijmeni": "Novák",
  "vek": 24,
  "adresa": {
      "ulice": "Božetěchova 13",
      "city": "Brno",
      "psc": "612 66"
  },
  "kontakty": [
      { "type": "email", "value": "jan@novak.cz" },
      { "type": "telefon",  "number": "123456789" }
  ]
}
```

Syntaxe je velmi jednoduchá a při správném formátování dobře čitelná. Objekty jsou uzavřeny ve složených závorkách, pole v hranatých. Klíč musí být vždy řetězec, dvojice klíč – hodnota jsou od sebe odděleny dvojtečkou a jednotlivé záznamy v poli i objektu čárkou. Bílé znaky se ignorují.

Extensible Markup Language (XML)

Je značkovací jazyk, pomocí kterého lze vytvářet textové řetězce nesoucí data[21]. Je relativně dobře čitelný i pro člověka a není ani na textovou formu zápisu zrovna úsporný. *PHP* disponuje funkcemi na zpracování *XML*, které ale nejsou příliš programátorsky přívětivé, takže se často používají dodatečné knihovny. Ani v *JavaScriptu* není vestavěná podpora. Nabízí mnohem více možností než *JSON*, například pokročilou specifikaci a kontrolu formátu dokumentu s využitím *DTD* souborů.

Comma-separated values (CSV)

Jedná se o jednoznačně nejstarší z uvedených formátů, je dokonce starší, než osobní počítače. Slouží k jednoduché textové reprezentaci tabulek. Jednotlivé buňky jsou od sebe odděleny právě čárkou, řádky prostým odřádkováním. Existuje mnoho variací, některé uzavírají obsah „buněk“ do uvozovek, jiné se liší oddělovacím znakem. Díky množství variant byl standardizován až v roce 2005[15], ale přesto je právě kvůli své jednoduchosti široce rozšířen právě v kancelářských aplikacích a účetních systémech.

Base64

Používá k reprezentaci dat jen 64 tisknutelných znaků. Každý ze znaků má přiřazenu šestibitovou sekvenci. Binární data rozdělí do sekvencí s délkou 6 bitů a k reprezentaci každé z nich použije odpovídající znak z překladové tabulky. Kódování se provádí po 3 bytech, protože nejmenší společný násobek 6 a 8 je 24. Pokud je sekvence kratší, doplní se výsledná reprezentace rovnítky. Používá se tam, kde očekáváme jen tisknutelné znaky, například přílohy elektronické pošty, *XML* dokumenty atd. Výslednou velikost zjistíme pomocí vzorce $\lceil n/3 \rceil * 4$, kde n je počet bytů zakódovaného řetězce. Největší nárůst je pro řetězec o 1 bytu a to o 300 %, nicméně už u 20 bytů je jen 40%. Toto chování je způsobeno právě kódováním po třech bytech, pokud je délka původních dat násobek 3, je nárůst jen o 1/3 (například už pro řetězec o délce 3) a čím je řetězec delší, tím je

4.1.7 jQuery

Je *JavaScriptová* knihovna, která značně zjednodušuje jeho použití[8]. Vylepšuje práci s objektovým modelem dokumentu tím, že podporuje selektory podobné těm z *CSS* a zrychluje tak přístup k jednotlivým objektům. Obsahuje spoustu užitečných funkcí pro manipulaci s objekty dokumentu nebo událostmi. Dokáže pro programátora zakrýt rozdíly mezi jednotlivými prohlížeči a ten tak může používat jednotné a mnohem jednodušší API, například u asynchronních požadavků. *JavaScript* má velkou škálu schopností, ale jejich použití je bez podobné knihovny velmi složité. Právě knihovny jako *jQuery* ho udělaly jednoduše použitelným i pro běžné vývojáře a posunuli ho tak o velký kus dopředu. Mimo to implementuje i jádro pro animace a usnadňuje manipulaci s *CSS*.

jQuery UI

Rozšiřuje *jQuery* právě o grafické prvky, vylepšuje jeho animační schopnosti a nabízí graficky jednotnou sadu prvků pro vytváření formulářů a jejich pohodlné vyplňování[9]. Obsahuje metody, které z běžných objektů udělají posouvatelné, řaditelné, umožní jim změnit velikost myší atd. Umožňuje například jednoduchou tvorbu tlačítek s ikonami, kalendář na výběr data, našeptávač a mnoho dalšího. Vše je navíc podrobně dokumentováno a vysvětleno na příkladech, takže není větší problém si funkce přizpůsobit nebo za pomoci jejích součástí další funkce vytvořit.

4.1.8 Java applet

`Java` je objektově orientovaný jazyk vytvořený firmou `Sun Microsystems` v květnu 1995[13]. Jeho hlavní výhodou je přenositelnost kódu. Stejná implementace je až na některá specifika spustitelná napříč různými platformami a hardware. Není potřeba ji nijak upravovat a dokonce ani znovu kompilovat. Tato vlastnost je zaručena skutečností, že zdrojové kódy se nepřekládají do binárního formátu, ale tzv. bytekódu, a na cílovém zařízení musí být nainstalována *Java Virtual Machine* (*JVM*), která jej interpretuje. Z toho vyplývají i jeho největší nevýhody. První z nich je paměťová náročnost *JVM*, jak název napovídá, jedná se virtuální stroj, který je spuštěn nad standardním operačním systémem a spoustu jím nabízených možností poskytuje duplicitně. Druhou z nich je rychlost, která je omezena tím, že ač je bytekód optimalizovaný, stále se jedná o interpretovaný jazyk. Tento neduh je částečně redukován některými mechanismy používané *Just In Time* (*JIT*) kompilace, která dokáže často znovupoužívané úseky kódu přeložit do binární podoby[14].

Jak již bylo řečeno, Java je velmi rozšířená a multiplatformní, proto se dočkala i modulu pro webové prohlížeče, který umožňuje spouštět webové applety přímo ve webové stránce. Ke spuštění appletu je tak nutné mít na počítači nainstalovanou *JVM*. *Java applety* umožňují velké množství funkcí, které prohlížeče nezvládají, například disponují nástroji pro pokročilou kryptografii nebo mají možnost přistupovat k periferiím počítače (čtečka čipových karet). Díky tomu ji využívá velké množství organizací, je nutná pro přístup do internetového bankovnictví některých bank a velké procento uživatelů ji tak má nainstalovanou. V případě webu *FakturyOnline.eu* je to podle *Google Analytics* dokonce 88,64 %. Značnou nevýhodou je jeho neexistence pro mobilní zařízení.

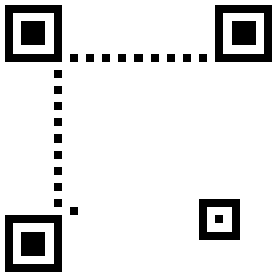
Java applety mohou požadovat dva stupně oprávnění: `sandbox` a `privileged`. V prvním případě jsou jejich schopnosti značně omezeny, mohou komunikovat jen se serverem

[8] http://jquery.com/
[9] http://jqueryui.com/

ze kterého byly spuštěny, nemají žádný přístup k souborům na počítači, tiskárnám atd. Pokud jsou spuštěny přes *Java Web Start* (*JNLS*), mohou omezeně přes speciální služby přistupovat k souborům, tisknout atd.[12] Bohužel s těmito funkcemi není příliš počítáno v ostatních Java třídách, například právě v kryptografických funkcích, a jejich použití je tak značně omezeno. Pokud applet běží v privilegovaném módu, má plný přístup k systému, stejně jako by to byla aplikace spuštěná z pevného disku počítače.

4.1.9 QR kódy

Jedná se o formát strojově čitelné grafické reprezentace dat[26]. Jde se o rozšíření klasického čárového kódu, který se nachází na velkém množství výrobků. Zatím co u něj byla data reprezentována kombinací různě širokých černých a bílých čar, QR kód se skládá z černých a bílých čtverečků, které mohou představovat několik různých druhů obsahu včetně binárního kódu. Původně byl určen pro sledování vozidla během jeho výroby, populární se stal až s nástupem chytrých mobilních telefonů. Ty totiž v drtivé většině obsahují fotoaparát, který může s vhodným software sloužit jako jednoduchá čtečka. Informaci reprezentovanou v kódu tak lze do telefonu jednoduše přenést. Opisování adresy webových stránek z billboardu je pro uživatele mnohem více zatěžující, než spuštění aplikace a zamíření telefonu.



Obrázek 4.1: Zarovnávací část QR kódu

Obrázek 4.2: Příklad QR kódu

Část teček slouží pouze pro určení orientace kódu samotného, taková šablona je vidět na obrázku 4.7. V těsném okolí je uložena informace o způsobu kódování obsahu. Kromě klasického bytového je možno použít například alfanumerické nebo jen číselné. Takováto kódování jsou vytvořena s ohledem na maximální úsporu místa, aby kód mohl reprezentovat co nejvíce obsahu pomocí nejmenšího možného počtu značek. Kdyby byl výsledný kód příliš jednotvárný, byl by pro zařízení špatně čitelný a existuje tak několik maskovacích vzorů, na jejichž místech se bity negují. Nejvhodnější maska se vybere podle předem stanovených metrik, jejichž cílem je zajistit co nejlépe čitelný kód. Kromě toho zpráva obsahuje nastavitelné množství redundantních znaků, pomocí kterých lze zprávu v případě poškození nebo nedokonalého načtení opravit.

4.2 Realizace

Pro vývoj aplikace byl zvolen inkrementální neboli přírůstkový model. Projekt je rozdělen na několik menších samostatně funkčních celků, pro které je postupně aplikován vodopádový model vývoje od návrhu až po testování. Až po dokončení jedné části se přejde na další.

Základem bylo přeuspořádání adresářové struktury. To původní bylo značně chaotické a kombinovalo různé přístupy. Soubory byly rozděleny do adresářů podle typu (asynchronní skripty, moduly webu, styly atd.). Uživatelským datům byl vytvořen speciální adresář, který

je nadále rozčleněn podle jejich druhu. Takovéto umístění umožňuje v budoucnu jejich jednoduchý přesun do jiného umístění, protože do adresáře lze v UNIXových systémech připojit bloková zařízení včetně disku. Navíc je jednodušší ochrana proti neoprávněnému přístupu, stačí totiž zakázat přístup jen na jednom místě.

4.2.1 Základ aplikace

Bylo potřeba vytvořit nové jádro aplikace, které by ale bylo zároveň kompatibilní s moduly původního systému, nejlépe tak, aby se vůbec nemusely upravovat. Původně probíhalo vložení modulu tak, že na patřičném místě šablony se vložil soubor, který provedl patřičné akce. Hlavní nevýhoda toho způsobu spočívá v nemožnosti ovlivnit již vypsaná data jako je například záhlaví webu.

PHP disponuje nástroji na kontrolu výstupního bufferu[17]. Pomocí funkce `ob_start()` lze zapnout bufferování výstupu, poté vložíme modul, on provede svou činnost a jím vyprodukovaný výstup si funkcí `ob_get_contents()` uložíme do vlastní proměnné. Poté výstupní buffer vyčistíme a pokračujeme na šablonu, jen v místě k tomu určeném vypíšeme výstup modulu z naší proměnné. Hlavní výhoda tohoto způsobu je, že už při generování stránky podle šablony máme dostupné proměnné, které modul vytvořil a ty tak mohou ovlivňovat vše od titulků stránky po načítané soubory JavaScriptu.

Databáze

Databáze v původním systému prošla kontrolou a kdybych ji navrhoval v současnosti, byla by její struktura více méně stejná. Rozhodně by se lišilo pojmenování některých sloupců a tabulek, ale to je věc, která kromě pohodlí programátora a přehlednosti nemá žádný praktický dopad na funkčnost. Chyběly v ní některé vyhledávací indexy, ale ty nebyl žádný problém doplnit. ale to zatím nezpůsobuje. Lze ji tedy využít v současné podobě a bude jen rozšířena o nové tabulky a sloupce. Některé sloupce se staly nadbytečnými, ale pro jistotu zatím nebyly odstraněny, kdyby byla data z nich někdy v budoucnu potřeba.

Správa chyb

Pracovat s chybami je potřeba napříč celým webem. Za tímto účelem byla vytvořena jedna třída, která zapouzdřuje práci s hláškami. Do proměnné `$h` se vytvoří instance třídy `Hlasky`, ta obsahuje metody pro přijetí hlášek a jejich výpis. Hlášky jsou rozděleny na 3 skupiny. Pro zprávy o úspěšném provedení operace, chyby uživatele a chyby serveru. O chybu uživatele se jedná pokud například uživatel zapomene vyplnit vstupní pole. O chybu serveru se jde v případě, pokud se například nepodaří uložit data do databáze. Je zřejmé, že taková chyba by neměla za normálních okolností vůbec nastat a proto je u nich nastaveno i automatické oznamování e-mailem na adresu správce systému. Zpráva kromě chyby samotné obsahuje i výpis aktuální paměti a případně i detailní informace o poslední chybě *MySQL*. Mimo to obsahuje funkce na výpis hlášek. Vrací přímo formátovaný HTML kód nebo pole, které lze dále zpracovat.

Výpisy

Klasických tabulkových výpisů je několik napříč celou aplikací. Už v původní aplikaci zvládaly stránkování nebo řazení podle jednotlivých sloupců. Bohužel všechny jejich funkce byly psány přímo ve stránce, úroveň abstrakce byla minimální a přehlednost kódu mizivá.

Nicméně i tak roky spolehlivě fungovaly, rozšíření (například o vyhledávání) jsem provedl přímo v nich. Jediným výpisem, který bylo potřeba zcela předělat bylo zboží. Rozhodl jsem se toho využít a pokusit se o nějakou přehlednější možnost výpisu. Vytvořil jsem tedy třídu *Table*, která zapouzdřuje vše potřebné pro vytvoření výpisu. Stačí z ní vytvořit objekt, vložit do něj *SQL* dotaz a nastavit, které sloupce a jakým způsobem se mají vypsat.

Pro výpis jsou typicky potřeba dva databázové dotazy. Jeden pro zjištění celkového počtu položek, druhý na vypsání aktuální stránky. Třídě stačí zadat jen jeden, o stránkování se postará sama, stejně tak si dotaz optimalizuje tak, aby při zjišťování celkového počtu od *MySQL* zbytečně nepožadoval data, ale jen počet položek. Díky konečnému automatu si poradí i se složenými dotazy. Sama si do databáze ukládá nastavení řazení pro konkrétního uživatele, umožňuje při přechodu mezi stránkami předávat libovolná data a má i další široké možnosti nastavení.

Komplikace nastala při nastavování výpisu řádku. PHP totiž neumožňuje ukládat funkce do proměnných a nebylo ji tedy možné funkci rozumně předat. Pouze zápisem `promenna()` se provede funkce, jejíž jméno je v proměnné uložené, což je jen omezená náhrada. Při předávání názvu proměnné by programátor musel kontrolovat, aby náhodou nepojmenoval výpisové funkce duplicitně a obecně se nejedná o příliš elegantní řešení. Druhou možností je vytvoření potomka třídy a dodefinování výpisové metody. Tohle už je relativně bezpečné řešení, ale opět relativně komplikované. Zvolena byla třetí varianta, a to použití funkce `eval`, která dokáže vykonat příkazy zapsané v řetězci. Pro komplikovanější problémy by bylo lepší zvolit druhou variantu, ale pro pouhý výpis řádku je toto řešení zcela dostačující.

Pokud se třída osvědčí, budou jí nahrazeny i ostatní výpisy. Přehlednost kódu je nesrovnatelně větší, což zamezuje chybám při jeho rozšiřování. Nakonec se ukázalo, že vytvoření zcela nového výpisu s její pomocí bylo rychlejší, než přidání vyhledávání do výpisu původního.

4.2.2 Implementace nového formuláře

Jednoznačně nejdůležitější částí aplikace je formulář pro tvorbu nové faktury. Díky zvolenému konceptu zahrnuje většinu ostatních funkcí portálu. Tato změna byla také nejvíce žádána, proto byla provedena jako první. Původní web byl psán procedurálně a veškerý kód potřebný k vytvoření faktury byl tedy zapsán přímo u zpracování dat z formuláře. U nové verze bude nutné faktury vytvářet přes více rozhraní (web, API) a proto je vhodné její vytváření oddělit. Bylo tedy třeba vytvořit samostatnou třídu, která obstará základní práci s fakturou a k ní webové rozhraní, které ji bude používat.

Třída na práci s fakturou

Má dvou skupin metod, podle jejich účelu. První slouží k nastavení dat o faktuře nebo jejich získání, druhé jsou výstupní.

Nastavovací a získávací metody Všechny vlastnosti třídy jsou soukromé, pro každou z nich je tedy potřeba metoda pro její nastavení a získání. Samozřejmostí je i uložení či načtení faktury z databáze. Třída samotná neobsahuje kontrolu platnosti vstupních dat, ošetřuje pouze možná bezpečnostní rizika, především chrání před *SQL injection*[2]. Tj. proti pravděpodobně nejrozšířenějšímu způsobu útoku na databázové servery, kterému se podrobněji věnuje kapitola 4.3.1.

Funkce kontrolující správnost dat nebyly umístěny přímo do této třídy, protože se liší podle vstupního rozhraní a také jsou potřeba i tam, kde se faktura vůbec nevytváří, například při ukládání uživatelů a zboží do databáze. Byl proto vytvořen samostatný soubor `checks.php`, který obsahuje kontrolní funkce pro jednotlivé části faktury. Každé z funkcí se předá odkaz na pole, obsahující data ke kontrole a objekt na zpracování chyb. Takto je jednoduše zajištěna znovupoužitelnost těchto funkcí, přičemž je nadále možné vytvářet specifické kontroly specifické pro daný vstup.

Výstupní metody Třetí skupinou jsou metody výstupní, tedy pro vytvoření a uložení *PDF* souboru nebo jeho odeslání e-mailem. Pro generování faktury se používá již zmíněná knihovna *tFPDF*. Různé jazykové mutace jsou řešeny souborem s překlady pro jednotlivé jazyky. Při výpisu se používají proměnné pole `$t`, do kterého se podle nastaveného jazyka uloží požadovaný překlad. Díky použití *UTF-8* není potřeba řešit žádné problémy s kódováním.

K odesílání zpráv e-mailem se nevyužívá žádná knihovna, pouze se poněkud nekonvenčně použije v *PHP* vestavěná funkce `mail`. Ta sice nativně neumožňuje přikládání příloh, ale umožňuje modifikaci hlavičky zprávy. Má 4 parametry: adresu příjemce, předmět, tělo zprávy a hlavičku. Pro naše účely tedy stačí sestavit si celou zprávu v *MIME*[11] formátu ručně, předat ji do hlavičky a jako tělo zadat prázdný řetězec. Formát *MIME* (*Multipurpose Internet Mail Extensions* tedy *Víceúčelové rozšíření internetové pošty*) umožňuje zasílat elektronickou poštou i jiný obsah, než jen prostý text. Do hlavičky zprávy jednoduše doplníme informaci, že jedná o MIME zprávu s přílohami a jaký jedinečný řetězec použijeme k oddělení jejích jednotlivých částí. Pomocí něj potom zprávu rozdělíme na několik částí, z nichž každá má vlastní hlavičku obsahující informace o tom, co a v jakém formátu je v ní uloženo. Teoreticky můžeme tuto část pomocí *MIME* opět rozdělit, což se nám může hodit pokud chceme uživateli nabídnout více formátů se stejným obsahem, například pokud je tělo právy k dispozici v čistém textu i *HTML*.

Implementace webového rozhraní

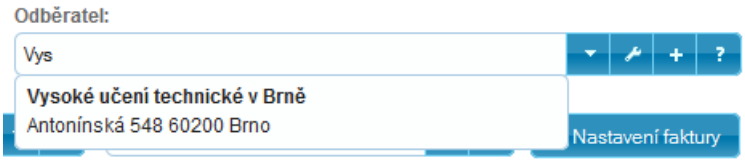
Byla provedena v *HTML*, *CSS* a *JavaScriptu* s využitím *jQuery UI* realizující vytvořené návrhy. *jQuery* nabízí mimo jiné možnost vytvářet přesouvatelná okna přímo ve webové stránce z jednotlivých bloků HTML kódu, což značně zjednodušilo zobrazování a skrývání jednotlivých částí formuláře. Mohl tak zůstat zachován koncept jednoho formuláře a původní možnosti nebyly nijak omezeny. Rozdíl oproti původnímu formuláři je ten, že nyní data se nyní asynchronně i ukládají, dříve se uložení provedlo až při vytváření faktury. Kontrola platnosti vstupních dat zůstala až na výjimky zachována na serveru při zpracování formuláře.

Aby byl formulář graficky jednotný, bylo nutné si vytvořit vlastní ekvivalent elementu `select`. Stačilo využít existující našeptávací plugin, jako zdroj mu nastavit pole obsahující nabízené možnosti a prvek, do kterého uloží výsledek. Vlastní alternativy byly vytvořeny také k funkcím `alert` a `confirm`. Barevně sladěné okno lze vyvolat jednoduchým příkazem obsahujícím hlášku. V případě potvrzení se v druhém parametru předá funkce, která se provede, pokud uživatel činnost odsouhlasí.

Ve formuláři jsou často různé prvky (vstupní pole, tlačítko, přepínač) těsně za sebou a je potřeba aby dohromady tvořily bezchybně jeden optický celek. Bohužel i přes použití zcela identických stylů se lišily jak velikostí, tak vertikálním odsazením, přestože ani vlastní debugovací nástroje prohlížečů žádné odsazení neukazovaly. Navíc v každém pro-

hlížeči byla situace trochu jiná, takže nešlo jednoduše použít korekci například relativní pozicováním. Situaci se podařilo vyřešit až použitím plovoucího zarovnání `float` místo původního `inline-block`.

Zdrojový kód stránky se skládá ze dvou částí. První nastavuje *JavaScriptové* proměnné, kontrolující splatnost faktury, možnosti zaokrouhlení, povolené jednotky, měny atd. Druhá obsahuje celý *HTML* formulář s předvyplněnými hodnotami. Ten se *JavaScriptem* zformátuje a z jeho součástí se vytvoří jednotlivá okna, která jsou ve výchozím stavu skrytá. Jak je formulář uspořádán lze jednoduše demonstrovat v části nastavení odběratele.



Obrázek 4.3: Výsek části formuláře pro nastavení odběratele

Na obrázku 4.3 je vidět, že v samotném formuláři jsou jen opravdu nejčastěji používané součásti. Textové pole, ve kterém je vidět aktuálně zvolený odběratel. Při zápisu do pole jsou okamžitě asynchronně našeptávání existující odběratelé odpovídající zadanému textu. Vpravo následuje tlačítko pro zobrazení těchto odběratelů. Je zde umístěno spíše pro přehlednost, vyhledávání totiž proběhne i pro prázdný řetězec (kdy jsou zobrazeni všichni) ihned po kliknutí do pole, nicméně tato schopnost není příliš obvyklá, zatímco šipka připomíná běžně známý element *select*. Za ní jsou umístěna tlačítka pro editaci a vložení nového odběratele. Obě dvě otevřou okno pro editaci. Při vytváření se všechny hodnoty vymažou, editační formulář zase obsahuje dvě tlačítka pro uložení. Jedno z nich uloží odběratele jako nového, druhé aktualizuje již existující záznam. Poslední je tlačítko pro nápovědu. Ta se načítá asynchronně podle klíče prvku ze skriptu obsahujícího pole textů a je zobrazena ve vlastním okně.

Všechny formuláře obsahují volbu *Storno*. Tu *jQuery UI* nijak neimplementuje, všechny změny se provádí přímo ve formuláři a nejsou tedy jednoduše vratné. Bylo tedy nutné tento problém vyřešit. Do *jQuery* jsem přidal funkce `pushValues()` a `popValues()`. První z jmenovaných se volá při otevření formuláře nad jeho obsahem. Projde všechny jeho potomky a pokud se jedná o vstupní prvek, jeho hodnotu si uloží. Jako úložiště použije *jQuery* funkci `data(klíč, hodnota)`, která umožňuje navázat data k nějakému prvku. Pokud uživatel okno zavře bez uložení, zavolá se funkce `popValues()`, která hodnoty z tohoto úložiště obnoví.

Všechna data, včetně tvorby faktury samotné, se ukládají asynchronně. Z patřičné části formuláře se hodnoty pomocí funkce `serialize()` exportují do řetězce a pomocí `$.post()` odešlou na server ke zpracování. Data se na server odesílají standardním *POST* požadavkem, server odpovídá ve formátu *JSON*. Vrátí informace o (ne)úspěchu požadavku včetně chybových hlášek. Ty jsou pak pomocí další přidané funkce `hlaska()` zobrazeny. Protokol komunikace je velmi jednoduchý. Všechny požadavky se odesílají skriptu `save.php`, k ukládaným hodnotám se přidá proměnná `action`, která nastavuje požadovanou akci. Odpověď server odešle v následujícím formátu:

```
{
  "id":43,
  "success":1,
  "chyby":[],
  "hlasky":["Údaje o~uživateli byly úspěšně uloženy"]
}
```

`id` obsahuje při ukládacích akcích ID položky v databázi. `success` informaci, zda akce uspěla. `chyby` obsahují pole řetězců s informací o tom, proč neuspěla a `hlasky` případně člověkem čitelné potvrzení úspěchu.

Serverová část Je velmi jednoduchá, využívá výše popsané (4.2.2) třídy pro tvorbu faktury. Pomocí příkazu `switch` vybere požadovanou akci, pokud je to potřeba vstupní data ošetří a naskládá je do očekávaných proměnných. Provede kontrolu vstupních dat pomocí funkce ze souboru `checks.php` a pokud uspěje, uloží data do databáze. Pokud byl požadavek na vytvoření faktury, tak místo přímého uložení vytvoří objekt pro tvorbu faktury, nastaví ho a nadále pracuje s ním.

ARES Pro zjišťování informací o dodavatelích je třeba komunikace s *ARES*. Jak již bylo zmíněno, ten poskytuje data v *XML* formátu, a tak by bylo možné, aby s ním komunikoval přímo uživatel. Ale kromě požadovaných dat je poskytována i spousta nepotřebných, navíc by bylo na straně klienta potřeba vyřešit zpracování *XML*. Proto zde server slouží jako prostředník, který data z databáze na žádost stáhne, vybere z nich jen ta důležitá a předá je v *JSON* formátu uživateli. Nicméně *ARES* má limit požadavků pro jednu *IP* adresu dotazů nastaven na 1000 dotazů v době od 8:00 do 18:00 a 5000 dotazů v době od 18:00 do 8:00 rána následujícího dne[10]. Pokud by tedy došlo k velkému nárůstu uživatelů, pravděpodobně by se zavedlo přímé dotazování.

4.2.3 Přesnost aritmetiky

Bylo potřeba zajistit, aby výpočty nebyly v žádném případě ovlivněny nedokonalostmi reprezentace desetinných čísel v plovoucí čárce. Nejjednodušším a hlavně nejspolehlivějším řešením je neprovádět výpočty ve dvojkové soustavě, ale přímo v desítkové. Počítače ji sice nativně nezvládají, ale dokáží ji simulovat. Náročnost výpočtů je sice mnohonásobně větší, ale množství prováděných operací není tak velké, aby to způsobilo nějaké komplikace. Pro tuto problematiku je na výběr dokonce z několika *PHP* modulů. K dispozici jsou dvě knihovny: *BC Math*[10] a *GNU Multiple Precision*[11]. Obě pro naše potřeby nabízejí všechny potřebné funkce mimo zaokrouhlení. Druhá jmenovaná je obecnou knihovnou napsanou v *C* a do *PHP* byla jen portována. *BC Math* byla vytvořena přímo pro *PHP*, čísla ukládá pouze v řetězcích a je celkově méně komplikovaná a přesto poskytuje vše potřebné, proto byla pro výpočty zvolena ona. Zaokrouhlení je relativně jednoduchá funkce, kterou nebude problém si napsat. Lze na ní názorně vidět, jak se s těmito funkcemi pracuje.

```
function bcround($strval, $precision = 0) {
  $zeros = str_repeat("0", $precision);
  if ($strval[0]=="-") return bcsub($strval, "0.{$zeros}5", $precision);
    else return bcadd($strval, "0.{$zeros}5", $precision);
}
```

[10] http://www.php.net/manual/en/book.bc.php
[11] http://www.php.net/manual/en/book.gmp.php

Má dva parametry, první obsahuje samotné číslo, druhý přesnost, na jakou má být zaokrouhleno. Jednoduše odečte (pokud se jedná o záporné číslo) nebo přičte číslo mající číslici 5 v řádu o jedna nižším, než na který zaokrouhlujeme. Sčítací či odečítací funkce zároveň odsekne nepotřebnou část řetězce, protože jí byla předána požadovaná přesnost výpočtu.

Dalším krokem byla úprava kalkulační metody třídy faktury, kde stačilo nahradit běžně používaná znaménka za tyto funkce.

4.2.4 Nastavení výpočtu DPH

Po prostudování platné legislativy bylo rozhodnuto o vytvoření čtyř možností nastavení výpočtu cen:

- *Zaokrouhlit částku* – způsoby zaokrouhlení výsledné částky faktury na celé koruny.
  - *Nezaokrouhlovat.*
  - *Výslednou bez danění* – výsledná částka se zaokrouhlí a vyrovnání se přidá na fakturu jako položka s nulovou sazbou DPH.
  - *Výslednou a zaokrouhlení zdanit* – provede se zaokrouhlení jednotlivých sazeb DPH na celé koruny, vyrovnání se zdaní a přidá na fakturu jako položka s patřičnou sazbou DPH.
  - *Výslednou a znovu vypočítat DPH* – provede se zaokrouhlení na celé koruny jednotlivých sazeb DPH, vyrovnání se přidá na fakturu jako položka s nulovou sazbou DPH. Ale v přehledu DPH provede znovu výpočet daně shora a zdaní tak celou částku. Pravděpodobně poslední známý pokyn Ministerstva financí České republiky uvádí jako správnou tuto variantu[12].
- *Zaokrouhlit DPH* – vypočtené DPH na faktuře bude zaokrouhleno na celé koruny.
  - *Nezaokrouhlovat.*
  - *Sečtené sazby DPH* – DPH pro jednotlivé sazby bude zaokrouhleno v přehledu.
  - *Jednotlivé položky faktury* – DPH u každé položky faktury bude zaokrouhleno na celé koruny.
- *Vypočítat DPH* – určí způsob výpočtu DPH.
  - *Připočíst k ceně (zdola: § 37/1)* – DPH se přičte k částce zadané do kolonky *Cena* podle zákona o DPH § 37/1.
  - *Vypočítat z ceny (shora: § 37/2)* – DPH se vypočítá z částky zadané do kolonky *Cena* podle zákona o DPH § 37/2.
- *V přehledu DPH* – určí způsob sestavení přehledu DPH.
  - *Sečíst položky faktury* – spočítá sumu všech položek faktury.
  - *Znovu vypočítat daň* – sečte pouze základy a znovu provede výpočet DPH. Při velkém množství průběžně zaokrouhlovaných položek faktury se mohou čísla značně lišit.

[12] Dostupný na http://www.financnisprava.cz/cs/dane-a-pojistne/legislativa-a-metodika/pokyny-d/casove-cleneni/2003/pokyn-d-253-1586

Implementace nebyla jednoduchá, protože voleb je velké množství a k implementaci se již používaly funkce z knihovny *BCMath*, které vyžadují prefixový zápis. Bylo potřeba pečlivě promyslet a vyzkoušet jejich možné kombinace, nicméně po programátorské stránce nic obtížného. Stačilo doplnit do funkce patřičné vlastnosti, k nim nastavovací a získávací funkce a upravit metodu pro výpočty.

4.2.5 Číselné řady

Systém původně uměl jen jednu číselnou řadu pro každého uživatele a ta se navíc opravdu musela skládat pouze z číslic. Ale některé subjekty mají více řad faktur, například pro různé typy podnikání a ty je nejpřehlednější odlišit právě nějakým znakem před či za samotným počítadlem. První způsob řešení, který každého napadne je vytvořit si pro číslo objekt, který se bude skládat z předpony, počítadla a přípony. Vytvořit patřičnou tabulku v databázi a ve formuláři při tvorby faktury nabízet na výběr z číselných řad. Toto byla moje první volba, kterou jsem implementoval a dostala se až do fáze testování, které bohužel dopadlo negativně. Některým uživatelů vadilo, že si už nemohou číslo ručně upravit, jiným zase nevyhovoval formát a nebyli schopni jej jednoduše změnit. Proto jsem byl nucen vymyslet jiné řešení, které by dokázalo rozšířit původní primitivní systém číslování o použití znaků a podařilo se mi vymyslet šablony. Jejich použití vypadá následovně.

Šablona může obsahovat libovolné znaky mimo `", *, <, >, [, ], =, +, /, \`. Ty byly vyřazeny pro případ, že například při ukládání nebo exportu se číslo faktury používá jako název souboru. Všechny ostatní platné znaky se ve výsledném čísle zobrazí tak, jak byly zadány. Výjimku tvoří pouze složené závorky { } a znaky v nich obsažené. Ty totiž budou nahrazeny podle jejich významu. Povolené jsou:

- `R` – Aktuální rok
- `M` – Aktuální měsíc
- `D` – Aktuální den
- čísla `1` až `9` – Počítadlo. Číslo určuje, kolik znaků mu bude vyhrazeno.

Například pro nejčastěji používanou řadu skládající se z roku a pořadového čísla faktury, kterému jsou vyhrazeny 3 místa (například `2014001`) je šablona {R3}. Pokud je třeba někam doplnit nějaký další znak, stačí jej dopsat vně závorek. Například označení skládající se z roku, měsíce a čtyřmístného čísla, které jsou oddělené písmeny (`R2014M10Č0004`) se zapíše jako `R{R}M{M}Č{4}`. Vše mimo složené závorky zůstane na místech a jejich obsah se nahradí podle seznamu výše.

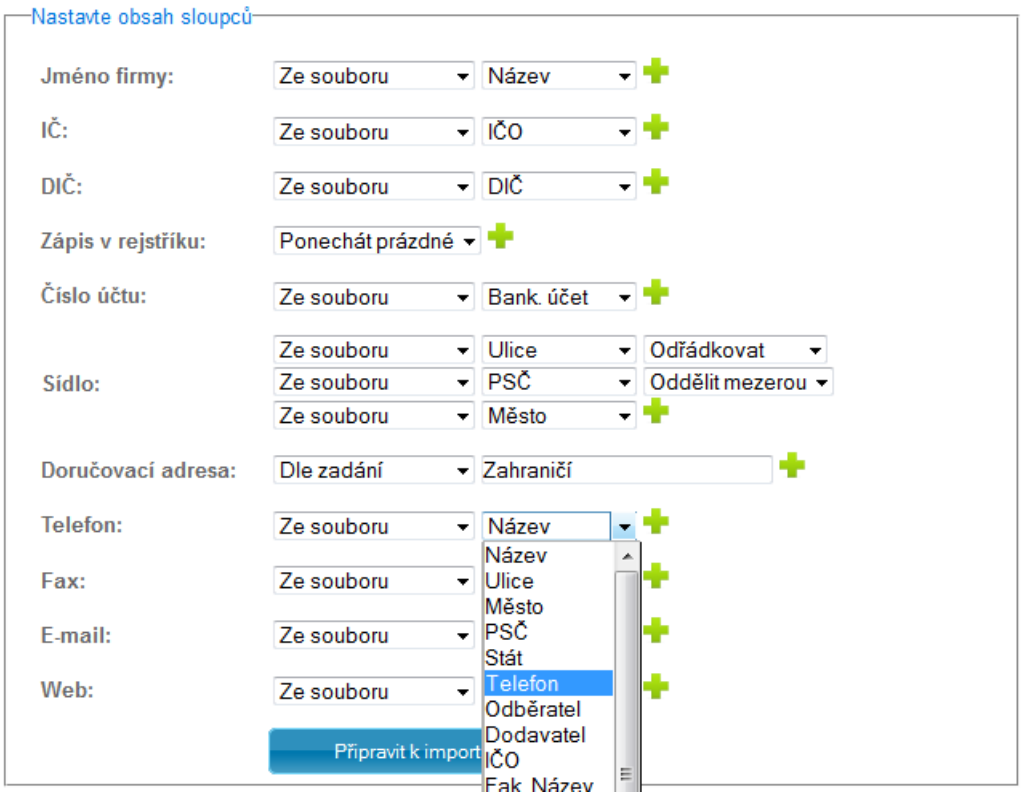
Aktuální hodnota počítadla se určuje stejně jako tomu bylo dříve (doklad s nejnovějším datem vystavení + 1) s tím rozdílem, že se berou v úvahu jen čísla dokladů odpovídající šabloně. Je tedy možné používat více číselných řad najednou. Jen je potřeba ohlídat, aby nebyly zaměnitelné. Problémem by byly například vzory `0{3}` a {4}, kde by u prvních 999 faktur druhé řady nebylo možné rozpoznat, zda nebyly vytvořeny podle prvního vzoru, měly by totiž na prvním místě také znak 0. Řešením je používat stejně dlouhá počítadla (`0{4}`, {4}) nebo nepoužívat statická čísla (`A{3}` a {4}), ostatní znaky zaměnitelné nejsou. Tento problém by si uživatel ale musel ohlídat stejně, dvě faktury se stejným číslem by se nelíbily ani finančnímu úřady.

Implementaci bylo složité vymyslet, ale jednoduché realizovat. Šablona se projde konečným automatem, který nahradí časové proměnné a poznamená si pozici a délku počítadla.

Pomocí těchto údajů sestaví regulární výraz pro *MySQL dotaz*. Ten jím poté dokáže rozpoznat, které faktury byly číslovány touto řadou, z nich vybere tu s nejnovějším datem vystavení. Z jejího čísla získá poslední hodnotu počítadla, navýší ji o 1 a pomocí ní sestaví výsledné číslo.

Při tvorbě faktury jsou pak počítadla nabízena rovnou s následující hodnotou a mimo to má uživatel možnost zadat nebo upravit číslo manuálně. Byly tak opravdu zachovány všechny možnosti původního formuláře a přesto se je podařilo rozšířit o požadovanou funkcionalitu.

4.2.6 Import adresáře

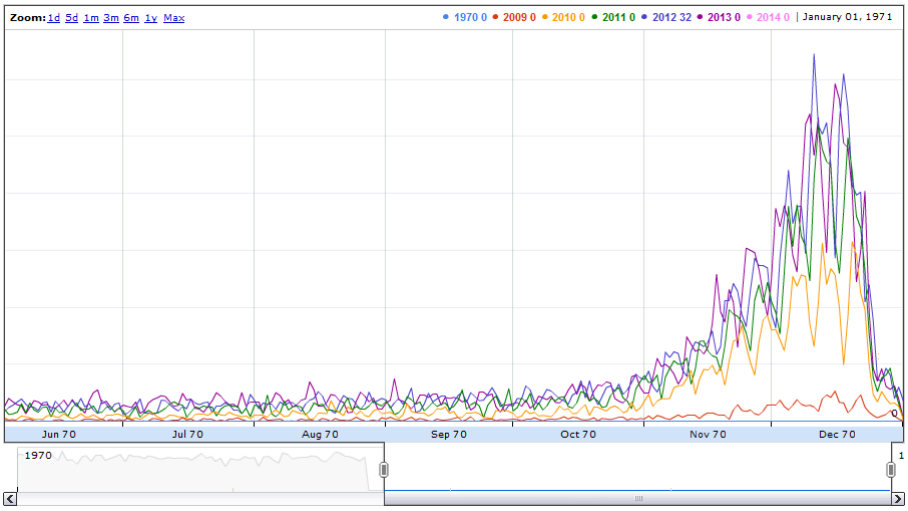


Obrázek 4.4: Nastavení importu

Nepodařilo se najít jeden dostatečně rozšířený formát, do kterého by mohla exportovat většina uživatelů. Proto byl zvolen přesně opačný přístup a to podpora všech. Jedinou podmínkou je, aby se jednalo o *CSV* a na prvním řádku bylo záhlaví. Na dalším uspořádání už nezáleží. Vstupní kódování a znak oddělující jednotlivé sloupce může uživatel nastavit. Uživateli se zobrazuje náhled souboru, aby bylo zřejmé, jak ho rozpoznal server. Má možnost podle záhlaví nastavit, který sloupec souboru odpovídá dotyčnému vstupnímu poli. Dokonce jich může spojit odřádkováním či mezerou více dohromady, zadat stejný text pro všechny položky nebo nechat pole zcela prázdné.

Dalším krokem je zobrazení všech položek, jak budou uloženy do adresáře, s možností editace a výpisem případných problémů. Jakmile tyto uživatel povtrdí, dojde k uložení záznamů. Pokud dojde k selhání jedné položky, bude ignorována.

4.2.7 Grafy



Obrázek 4.5: Ukázka grafu

Pro lepší vizualizaci časového vývoje příjmů byly do aplikace přidány grafy. Zatím pouze právě ty časové. Zobrazují vývoj počtu vydaných faktur a denních příjmů v průběhu roku. Pro generování grafů bylo zvoleno *Google Chart API*[13], které umožňuje jednoduše a kvalitně vizualizovat data. Stačí načíst knihovnu, vytvořit v *JavaScriptu* vstupní pole hodnot a inicializovat graf.

4.2.8 Párování plateb

Pro uživatele s větším počtem faktur by bylo výhodné, kdyby nemuseli označovat faktury jako zaplacené ručně. Tuto činnost lze zautomatizovat pomocí strojově čitelných výpisů z banky. Skript výpis projde a podívá se, zda variabilní symbol a částka platby nesouhlasí s nějakou zatím neuhrazenou fakturou.

Pro import z banky se používají *ABO* výpisy[14], jejichž podporu má většina finančních ústavů nabízejících podnikatelské účty. Jedná se o jednoduchý formát, kdy všechny platby jsou reprezentovány stejně dlouhým řetězcem a odděleny řádkováním. Každá část záznamu má přesně určený význam, pro zjištění potřebných údajů tedy stačí načíst patřičný podřetězec a s ním nadále pracovat.

4.2.9 API

Slouží pro strojové vytváření faktur, bude využíváno například různými e-shopy. Příliš se nebude lišit od standardního ukládání faktury, jediným rozdílem bude komunikační protokol a způsob autentizace. Uložené nastavení se samo načte a není třeba ho při každém požadavku odesílat znovu. Mimo to bude umět zobrazovat uložené faktury.

[13] Dostupné na https://developers.google.com/chart/

[14] Specifikace například na http://www.csas.cz/static_internet/cs/Obchodni_informace-Produkty/Prime_bankovnictvi/Spolecne/Prilohy/ABO_format.pdf

Protokol

Komunikace probíhá přes HTTP, s doporučenou možností využít SSL. Zprávy mezi serverem a klientem jsou předávány ve formátu *JSON*. V tomto formátu se předává i samotný požadavek. Běžně se pomocí *POST* požadavku předávají data ve formátu `application/x-www-form-urle` což znamená, že tělo požadavku obsahuje data zakódovaná stejně jako by byla v *URL*. *PHP* je umí standardně zpracovat do proměnné `$_POST`. `application/json` není takto podporován a je potřeba ho zpracovat manuálně. To naštěstí není žádný problém, protože k tělu požadavku můžeme přistupovat jako k souboru.

```
1  $telo=file_get_contents('php://input');
2  $data=json_decode($telo);
```

Nyní máme k dispozici vše potřebné, stačí data načíst do patřičných proměnných, provést kontroly a případně vytvořit fakturu, podobně jako u klasického uložení. Pokud je v odpovědi požadována faktura v *PDF*, zakóduje se její obsah pomocí algoritmu *Base64*, aby obsahoval pouze tisknutelné znaky a standardně se odešle jako součást *JSON*.

Autentizace

Autentizační údaje musí být trvale uloženy ve skriptu, proto by nebylo dobré používat stejné přihlašovací údaje jako ke standardnímu přístupu do systému. Také není důvod standardně povolovat použití API každému uživateli, většina ani neví, k čemu je dobré. Uživatelé tedy mají možnost zapnout si ho v nastavení, při aktivaci se jim na e-mailovou adresu zašle náhodně vygenerovaný autentizační klíč.

Přestože je dovolena komunikace přes *SSL*, není žádná záruka, že ho všechny klientské systémy budou podporovat. Proto ani autentizace nebude probíhat prostým zasláním dvojice *IČO* a klíč pro přístup k API. Pro přihlášení si musí uživatel vygenerovat ještě náhodný řetězec dlouhý minimálně 10 znaků a do proměnné uložit aktuální UNIXový čas. Tyto údaje se spojí do řetězce společně s přihlašovacím klíčem a z výsledného řetězce se spočítá kontrolní součet algoritmem MD5. Je potřeba dodržet uvedené pořadí.

```
1  $klic = md5($time.$api_auth.$salt);
```

Sůl a čas se pošlou v požadavku, aby server mohl klíč také spočítat a porovnat se zaslaným. Čas se musí lišit maximálně o 5 sekund od aktuálního času serveru, aby ho v případě odposlechnutí nebylo možné použít opakovaně. Server si navíc použité soli ukládá a během 24 hodin je není možné opakovaně použít.

Třída pro použití API

Uživatelé dostali k dispozici třídu pro jazyk *PHP*, která za ně obstarává používání samotného API. V konfiguračním souboru nastaví svoje *IČO*, autentizační klíč a zda si přejí používat *SSL*. Poté si jen vytvoří objekt, nastaví mu požadované parametry a nemusí se zatěžovat implementací komunikace s API. Třída využívá ke komunikaci funkci `fsockopen` a požadavek sestavuje ručně. Klade tak minimální požadavky na server, na kterém bude spuštěna.

4.2.10 Podepisování faktur

PDF dokumenty je možné opatřit digitálním podpisem, který verifikuje, že byla skutečně vytvořena a odeslána jejím vydavatelem. V případě PDF se do dokumentu doplní objekt

obsahující digitální podpis a do katalogu se doplní informace o to, že je dokument podepsán a reference na patřičný objekt.

Jedna z použitých tříd (`TCPDF`) podepisování standardně umí. Problémem je, že k podepsání dokumentu je potřeba certifikát obsahující soukromý klíč, který by bylo nutné alespoň dočasně nahrát na server. A certifikát umožňuje komukoli, kdo ho drží, se věrohodně vydávat za dotyčnou osobu. Uživatelé by tedy neměli být ochotni tento certifikát jen tak komukoli poskytovat a i pro správce serveru by přítomnost takto důležitých dat nebyla příjemná a dělala z něj lákadlo pro různé útočníky. Bylo tedy několik variant, jak tento problém vyřešit.

Zaheslované klíče Na serveru by byly uloženy pouze zašifrované klíče, heslo by musel majitel certifikátu při podepisování vždy znovu zadat. Nicméně i tak by existoval okamžik, kdy jsou všechny potřebné údaje v jednom okamžiku na serveru, i když heslo by bylo jen v paměti. Uživatelé by museli uvěřit, že heslo opravdu není skladováno a server by se musel velmi důkladně kontrolovat, zda nějaký útočník neobjevil skulinku v zabezpečení.

Podepisování přímo v prohlížeči Při této variantě bychom se museli spolehnout na to, co nám prohlížeče nabízí, tedy především *JavaScript*. *Internet Explorer* nabízí *CAPICOM*, což je *ActiveX* modul, který přímo v prohlížeči zpřístupňoval *Microsoft Cryptographic technologies*, které nabízely potřebné funkce[8]. Jeho podpora již ale byla ukončena a ve Windows 7 standardně není[9]. Chrome a Safari čekají až organizace *W3C* vydá *DOMCrypto* jako standard, zatím je jen ve fázi návrhu[22]. Mozilla Firefox už má podporu implementovanou jako `window.crypto`, který nabízí vše potřebné, ale bohužel podpora v jednom prohlížeči není dostačující[4].

Využití modulu prohlížeče Moduly jsou v prohlížečích od toho, aby rozšiřovali jeho schopnosti. Nejrozšířenější je *Adobe Flash*, který je ale zaměřen především na multimédia. Následuje Java, která v tomto ohledu poskytuje vše potřebné a její rozšíření mezi uživateli je velké, navíc ji lze do libovolného prohlížeče doinstalovat. Vše se provádí přímo na klientském počítači, certifikát jej nikdy neopustí. Díky kompilaci do bytekódu si mohou opatrnější uživatelé ověřit, že žádná část jejich certifikátu opravdu za žádných okolností neopustí jejich počítač. Navíc applet odesílá *HTTP* požadavky prostřednictvím prohlížeče, takže není potřeba speciálně řešit autentizaci, cookies zůstanou zachovány. Zvolena byla tato metoda.

Java applet

Se od normální Java aplikace liší minimálně. Jako rodič základní třídy se použije `JApplet`, jinak je vývoj téměř shodný. Výsledný produkt je *JAR* soubor. Do stránky se dříve vkládal pomocí speciálního tagu `<applet>`, ten již ale není v *HTML5* podporován a je doporučeno používat modernější univerzální `<object>`. V obou případech je vložení velmi jednoduché.

```
1  <applet code="Signer.class" archive="Signer.jar"
2   width="740" height="400"></applet>
3  <object codetype="application/java" classid="java:Signer.class"
4   archive="Signer.jar" width="740" height="400"></object>
```

Applet samotný je velmi jednoduchý na ovládání, jak je i vidět na obrázku 4.9. Uživatel vloží certifikát, zadá heslo (pokud existuje) a klikne na podepsat. Applet si ze serveru stáhne faktury určené k podepsání a vytvoří se jejich digitální podpisy, které se odešlou zpět

Obrázek 4.6: Podepisovací applet

na server. Bohužel po nedávných incidentech nelze v prohlížeči standardně vůbec spouštět applety nepodepsané důvěryhodným certifikátem. Pokud applet podpis nemá, applet se vůbec nespustí a uživatel uvidí jen chybovou hlášku. Takovýto certifikát stojí tisíce korun a nevlastním ho. Uživatel by si tedy musel buď přidat certifikát do důvěryhodných pro podepisování software nebo přidat web na seznam výjimek, kde je možno spouštět nepodepsané applety. A i poté si vyžádá upozornění, zda jej opravdu spustit. Přidání výjimky je jednodušší, proto je na webu umístěn návod jak to udělat. Pokud bude o službu zájem, certifikát zakoupím.

Serverová část K připravení faktury k podpisu se využívá knihovna `TCPDF`. Ta standardně podepisování dokumentů umožňuje. Pokud je příznak na podepisování aktivní a je načten certifikát, při generování PDF se vygeneruje i podpis a vloží se do dokumentu. Vytvořil jsem tedy potomka třídy *TCPDF* rozšířeného o upravenou výstupní funkci. Ta připraví dokument k podpisu a takto modifikovaný dokument vrátí jako řetězec a vytvoří dva dočasné soubory obsahující část dokumentu před a za podpisem. Tyto činnosti se vykonávají až v okamžiku, kdy applet o soubory požádá. Dokument k podpisu je tedy zakódován pomocí *Base64* a vložen do *JSON* objektu, který je odeslán appletu. Pokud je dokumentů k podpisu více, pošlou se všechny zároveň. Applet vygeneruje podpis, opět ho zakóduje pomocí *Base64* a odešle zpět na server. Ten podpis rozbalí, převede ho do hexadecimální reprezentace, doplní zleva nulami na očekávanou délku, vloží do dokumentu získaného z dočasných souborů a uloží. Výsledkem této operace je podepsaný PDF soubor.

4.2.11 Hromadné operace s fakturami

Do současnosti bylo nutné všechny operace s fakturami provádět jednotlivě, přičemž pro některé úkony je to zbytečnost. Jedná se především o hromadné exporty. Byly implementovány dva druhy exportu — do formátu *ZIP* ke stáhnutí do počítače a spojení do jednoho *PDF* pro tisk. Mimo to byla přidána možnost faktury hromadně stornovat a označit jako zaplacené. Ve výpisu se v prvním sloupci každého řádku zobrazuje zaškrtávátko, pomocí kterého lze faktury vybrat. Pokud se má operace provést se všemi vypsanými fakturami, existuje na posledním řádku možnost *Vybrat vše*, která hromadně vybere všechny řádky zobrazené ve výpisu.

Export do *ZIP* využívá *PHP* modulu *php_zip*. Ten umožňuje jednoduše vytvořit archiv, přidat do něj soubory a uložit ho. Jediná jeho nevýhoda spočívá v tom, že nedokáže vypsat výsledek do výstupního bufferu. Je tedy nutné použít dočasný soubor, který je na konci

přečten do bufferu a smazán.

Sloučení do PDF využívá kombinace knihoven *TCPDF* a *FPI*. *FPI* je vlastně potomkem *TCPDF*, který rozšíří o importní metody. Pro naše potřeby byla ještě rozšířena o funkci `concat()`, která obsahuje funkce pro přidání souboru. Přestože ji původní knihovna neobsahuje, stačí její ostatní funkce vhodně použít. Prvně se připojovaný soubor nastaví jako vstupní, tím se získá počet jeho stránek a pro každou z nich je potřeba vytvořit novou stránku v dokumentu a nastavit jí jako šablonu právě připojovanou stranu.

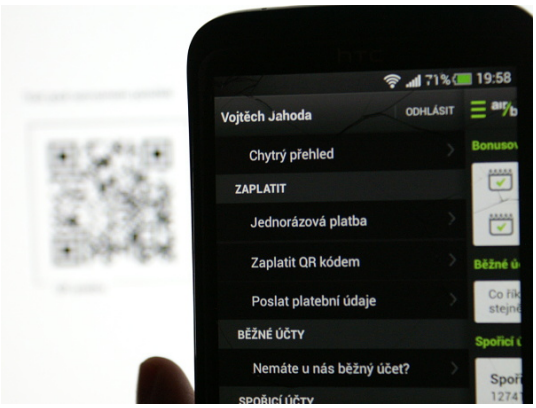
Ostatní hromadné operace nejsou nijak složité, typicky stačilo upravit *MySQL* dotaz, aby se vztahoval na více položek zároveň.

4.2.12 Platební QR kódy

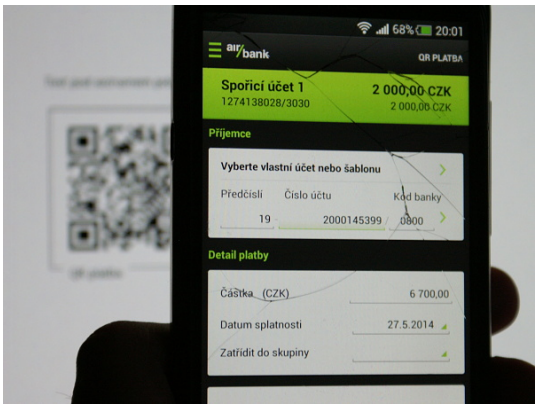
Oficiálně označovány jako *Short Payment Descriptor*, zkracováno jako *SPAYD* nebo *SPD*. Využívá QR kódy k ukládání informací potřebných k provedení platby, uživatel pak nemusí opisovat a kontrolovat údaje z faktury, ale stačí mu mobilní aplikaci své banky načíst takto vytvořený QR kód[24]. Jejich formát je v zásadě jednoduchý a byl inspirován stále populárním formátem na zasílání elektronických vizitek *vCard*. Může obsahovat několik předem daných údajů, které jsou od sebe odděleny symbolem `'*'`. První specifikuje, že se jedná o platební kód, druhý značí jeho verzi. Následují dvojice klíč – hodnota, které jsou od sebe odděleny dvojtečkou. Platební sekvence pro odeslání 400,40Kč na účet 19-2000145399/0800 s variabilním symbolem 273 tedy vypadá takto:

```
SPD*1.0*ACC:CZ6508000000192000145399*AM:400.40*CC:CZK*X-VS:273
```

Zdánlivě primitivní, ale přesto že se jedná o formát vymyšlený českou firmou a mimo Českou a Slovenskou republiku se (alespoň zatím) nepoužívá, požaduje číslo v mezinárodním formátu, také označováno jako *IBAN*. To sice zvyšuje jeho možnosti na rozšíření do zahraničí, ale u nás se tento formát u vnitrostátních plateb běžně nepoužívá a většina uživatelů tak ani neví, že ho jejich účet také má. Jedná se totiž jen o jiný zápis. České číslo účtu je složeno maximálně šestimístného předčíslí účtu, které je odděleno pomlčkou od samotného čísla účtu s délkou do deseti znaků. Za nimi následuje lomítkem oddělený kód banky. IBAN nepoužívá žádná oddělovací znaménka, místo toho má všechna čísla zleva doplněna nulami na jejich nevyšší možnou délku. Skládá se z dvou znaků pro kód země, dvou kontrolních číslic, čísla banky a předčíslí a čísla účtu.



Obrázek 4.7: Volba platby v mobilní aplikaci



Obrázek 4.8: Naskenovaný kód

Pro kontrolní čísla se používá algoritmus *Modulo 97*. Pokud vezmeme část obsahující kompletní číslo účtu, doplníme za ně číselnou reprezentaci státu, kontrolní číslice a celé takto získané číslo celočíselně vydělíme 97, musí nám vyjít 1, jinak číslo není platné.

Pokud bude mít uživatel *IBAN* zadaný, použijeme jej i pro generování platebního kódu. Pokud ho nezadal, můžeme ho vygenerovat z čísla účtu. Rozdělit číslo účtu na jednotlivé části a doplnit ho nulami není větší problém a i kontrolní součet lze dopočítat. Sestavíme číslo pro kontrolu, kde místo kontrolních číslic použijeme nuly a celočíselně vydělíme 97. Kód české republiky je 1235, pro již uvedené 19-2000145399/0800 tedy $080000001920000145399123500 \mod 97 = 33$. Kontrolní číslice získáme tak, že výsledek této operace odečteme od 98, tedy $98 - 33 = 65$. Tedy stejný výsledek, jako ve výše uvedeném příkladě. Pro výpočet musíme použít již zmíněnou knihovnu `BCMath`, protože *PHP* standardně s takto vysokými čísly pracovat nezvládá.

Pro generování QR kódu samotného pak byla použita knihovna *phpqrcode*, která je schopna vytvořit obrázek obsahující kód z libovolného textu[25]. Rámeček s označením doporučovaný grafickým manuálem je nakreslen přímo v PDF.

4.3 Zabezpečení

Důraz na důkladné zabezpečení aplikace musí být v dnešní době nedílnou součástí návrhu a implementace každé webové služby. Systém spravuje velké množství osobních údajů a citlivých dat různých subjektů a disponuje tak velkým množstvím aktiv. K zajištěné bezpečnosti informačního systému je potřeba zajistit důvěrnost, ochranu proti neoprávněné modifikaci informace a dostupnost[7].

4.3.1 Důvěrnost a ochrana proti neoprávněné modifikaci

Je potřeba zajistit, aby nikdo kromě oprávněného uživatele nemohl získat přístup k jeho datům nebo je upravovat. Tyto dva cíle byly spojeny dohromady, protože v našem případě opatření pro zajištění důvěrnosti zároveň zamezí i modifikaci dat. Ta jsou umístěny na Linuxovém virtuálním serveru a existují jen dva způsoby, jak se k nim dostat. První je přes *SSH*, druhý je přes webovou aplikaci. Žádné jiné služby nejsou na serveru dostupné. *SSH* umožní plný přístup k datům na server, ale přístup k němu mají pouze správci. Webová aplikace zpřístupňuje data uživatelům systému, a musí tak zajistit, aby se pomocí ní nebylo nijak možné dostat k cizím datům.

Bezpečnostní rizika a opatření

Tato část textu popisuje identifikovaná bezpečnostní rizika, jejich možný dopad a jaká opatření byla učiněna aby k nim buď vůbec nedošlo, nebo aby byl jejich dopad alespoň co nejvíce omezen.

Uhodnutí přístupových údajů Pro přihlášení k *SSH* je povolena autorizace pouze pomocí soukromého klíče, což tuto možnost prakticky eliminuje. Pro přihlášení k webové aplikaci se používá dvojice *IČO* a heslo. Pokud by se jednalo o cílený útok na některého z uživatelů, je pravděpodobné, že by útočník *IČO* znal a musel by zjistit jen heslo. K tomu by mohl použít slovníkový útok, kdy by nějakým programem systematicky zkoušel běžně používaná hesla. Jako opatření se veškeré neúspěšné pokusy o přihlášení ukládají do data-

báze a je jich dovoleno maximálně 10 za 24 hodin. To útočníkovi takovouto snahu značně zkomplikuje. Uživatelé jsou navíc při registraci vybízeni k volbě dostatečně složitého hesla.

Neoprávněné získání autentizačních údajů Jedná se o neoprávněné získání přihlašovacích údajů nebo cookie, která autentizuje uživatele po jeho úspěšném přihlášení, při jejich odesílání na server. Komunikace se serverem probíhá přes *HTTP*, takže pokud se případnému útočníkovi podaří nějakým způsobem zachytit komunikaci mezi uživatelem a serverem, mohl by tyto údaje získat. Jako protiopatření je na serveru aktivní *HTTP over SSL*, které před započetím samotné komunikace vytvoří šifrovaný tunel, kterým jsou data přenášena. Tento způsob komunikace je nastaven jako výchozí. Podvržení identity serveru je ztíženo tím, že disponuje *SSL* certifikátem podepsaným důvěryhodnou autoritou, u které může uživatelův prohlížeč ověřit identitu serveru.

U cookies lze nastavit, že je má prohlížeč odesílat pouze v šifrovaném spojení a riziko tak nehrozí ani když uživatel zadá adresu bez `https`. Aby se při použití nešifrovaného spojení uživateli „neztratilo“ přihlášení, nastaví se nijak neomezená cookie s logickou informací, zda je uživatel přihlášen. Pokud server zjistí nepřihlášeného uživatele na nezabezpečeném spojení, automaticky uživatele přesměruje na spojení zabezpečené a pomocí autentizační cookie ověří, zda je skutečně přihlášen.

Lidské selhání V tomto případě se jedná o vyzrazení přístupových údajů přímo jejich vlastníkem. Ať vědomě, například nějakému podvodníkovi, nebo bez jeho vědomí, například krádeží či ztrátou. U správců je riziko sníženo tím, že jejich certifikáty jsou šifrované heslem. Chování uživatelů nelze příliš ovlivnit, v případě vědomé ztráty si mohou přístupové heslo změnit.

SQL Injection Jedná se o útok na databázi s cílem získat z ní data nebo je nějak modifikovat. Spoléhá na to, že do SQL dotazů se často vkládají data, které vložil uživatel. Sestavení dotazu sloužícího k přihlášení uživatele může proběhnout například takto například takto:

```
mysql_query("SELECT * FROM uzivatele WHERE ic='$ico' AND heslo=MD5('$heslo')");
```

Pokud uživatel zadá do pole pro *IČO* `"12345678' AND ('"` a pro heslo `"heslo') OR 1=1 OR '"`, bude výsledný dotaz odeslaný databázovému systému vypadat následovně:

```
SELECT * FROM uzivatele WHERE ic='12345678' AND
                              ('' AND heslo=MD5('heslo') OR ''='');
```

Tedy nejen že útočníkovi umožní přihlásit se do systému, ale dokonce si může zvolit, za kterého uživatele. Správně napsané sestavení dotazu může vypadat takto:

```
mysql_query("SELECT * FROM uzivatele WHERE
                     ic='"mysql_real_escape_string($ico)".'
                     AND heslo='".md5($heslo)."'");
```

Funkce `mysql_real_escape_string` ošetří vstup tak, že přidá zpětná lomítka před všechny znaky, které by *MySQL* mohlo brát jako řídící. Výstup funkce *MD5* není třeba ošetřovat, protože vždy vrátí bezpečný alfanumerický řetězec. Stejně tak jsou bez problémové jednoznačně číselné datové typy, ale všechny vstupní řetězce je třeba důkladně ošetřit.

Cross site scripting (XSS) Jedná se o podobný typ útoku jako *SQL Injection*, opět se spoléhá na použití neošetřeného vstupu od uživatele. Tentokrát ale nejde o jeho vložení do SQL dotazu, ale přímo do stránky samotné. Takto vložený řetězec může obsahovat například *JavaScriptový* kód, který tímto získá přístup k obsahu webové stránky a jejím cookies, které může odeslat útočníkovi. Většina prohlížečů podporuje u cookies direktivu *HttpOnly*, která zajistí, že tato cookie nebude *JavaScriptu* vůbec přístupná. I tak má skript přístup k obsahu stránky samotné. Je spuštěn s identitou uživatele a jím odeslané požadavky tedy obsahují *cookies*, může si tak pomocí *AJAX* metod postahovat obsah citlivých URL a předat ho útočníkovi. Proto je nutné všechny vypisované vstupy důkladně ošetřit, aby nemohly obsahovat nic nebezpečného. V drtivé většině případů se od nich očekává pouze text. K tomu v *PHP* složí funkce `htmlspecialchars`, která všechny takovéto znaky převede na HTML entity. Je lepší ji použít s flagem `ENT_QUOTES`, který zajistí, že budou převedeny i jednoduché uvozovky[19].

4.3.2 Dostupnost

Tento bezpečnostní cíl zajišťuje, že uživateli nebude neoprávněně odepřen přístup k jeho datům. Existuje několik scénářů, jak by k takovéto situaci mohlo dojít.

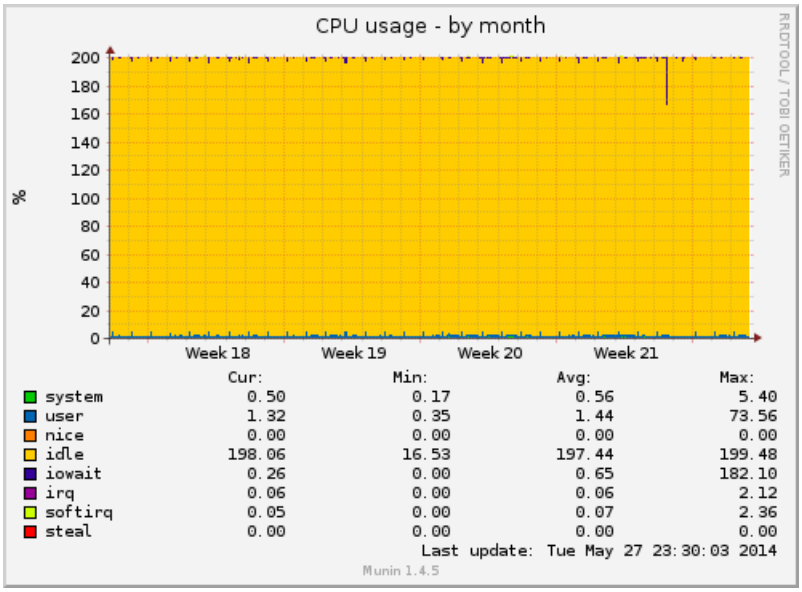
Zapomenuté přihlašovací údaje Docela často se vyskytující scénář, kdy uživatel zapomene svoje heslo. Přihlašování pomocí identifikačního čísla má tu výhodu, že jej lze jen velmi těžko zapomene a případně si ho může jednoduše zjistit. Nicméně zapomenout heslo může a proto musela být uživatelům dána možnost jeho obnovy. Děje se tak pomocí elektronické pošty. Uživatel na webu na základě svého identifikačního čísla požádá o nové heslo. Ale zjistit něčí *IČO* není žádný problém a kdokoli cizí by tak mohl komukoliv změnit heslo. Sice by ho tím nezískal, ale uživatel by si ho stejně musel změnit zpět. Proto je prvně na klientův e-mail zaslán mail s odkazem na potvrzení žádosti a až po jeho načtení je mu vygenerováno a na e-mail zasláno nové heslo.

Nedostupnost serveru Může se stát, že vlivem nějaké neočekávané události či softwarové chyby dojde k nedostupnosti portálu. Server je umístěn v serverovně, která se stará o jeho konektivitu, napájení i případnou výměnu problémového hardware. Server je nastaven tak, že je schopen sám po restartu automaticky bez lidského přičinění naběhnout do plně funkčního stavu. Na nás tedy zbývají softwarové problémy. Je aplikována konzervativní politika aktualizací, kdy je software sice pravidelně aktualizován, ale jsou delší dobu udržovány starší verze software a instalují se bezpečnostní záplaty. Tím lze omezit riziko zatím nenalezených chyb v nových částech kódu nebo nekompatibility některých novinek se současnou konfigurací a ostatním software.

Pro plynulý běh systému je nutný dostatečně dimenzovaný hardware, aby při případném neočekávaném nárůstů uživatelů nedošlo k jeho přetížení. Jak je vidět na grafu **??**, rezerva ve využití serveru je opravdu velká.

Důležité je také se o případných výpadcích zavčas dozvědět. Proto je server neustále monitorován a v minutových intervalech je testována dostupnost *HTTP* portu. Pokud dojde k výpadku, správci jsou okamžitě informováni SMS zprávou a e-mailem a mají tak možnost velmi rychle reagovat.

Ztráta dat Dalším způsobem, jak mohou být uživateli odepřena data je v důsledku havárie serveru samotného. Poskytovatel za data žádným způsobem neručí a zajištění zálohování



Obrázek 4.9: Graf využití procesoru serveru

dat je tedy na nás. Je potřeba zálohovat datové úložiště a databázi. K zálohování se používá program `rsync`, který komunikuje přes SSH a dokonale s minimálními nároky dokáže provádět inkrementální zálohy[23]. *MySQL* provádí jednou za den kompletní export databáze, který se ukládá do zálohovaného umístění. Ale export celé databáze je relativně náročný proces, proto se provádí v noci při minimálním vytížení serveru. Ale výpadek může nastat i v průběhu dne a například k večeru by už množství ohrožených dat bylo značné. Server má proto zapnutý binary log. To znamená, že veškeré příkazy činící nějakou modifikaci databáze se okamžitě ukládají do tohoto logu a je možné pomocí těchto souborů kdykoli obnovit aktuální stav databáze. Tyto soubory společně s veškerými daty na serveru každé 3 minuty synchronizují se záložním úložištěm.

Kapitola 5

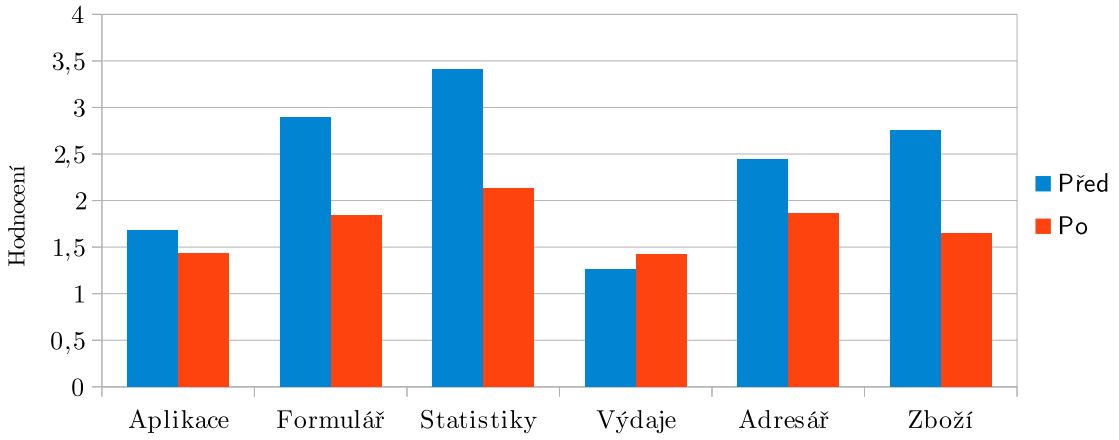
Analýza dopadu změn

5.1 Reakce uživatelů

Změny byly vždy přednostně předány k vyzkoušení testovací skupině. To zajistilo spolehlivé testování v širokém spektru prohlížečů i operačních systémů. Většina změn byla hodnocena příznivě, jedinou výjimkou byly již zmíněné číselné řady, které byly po vyhodnocení zpětné vazby zcela přepracovány do podoby, která už si opět od uživatelů získala pozitivní ohlas. Nakonec byl uživatelům předložen dotazník podobný tomu před započetím vývoje. Průzkum dopadl následovně:

Otázka	**Před změnami**	**Po změnách**
Jak jste spokojeni s aplikací jako celkem?	1,68	1,43
Jak jste spokojeni s těmito součástmi?		
Formulář na tvorbu faktury	2,89	1,84
Statistiky	3,41	2,13
Správa výdajů	1,26	1,42
Adresář	2,44	1,86
Zboží a služby	2,75	1,65

Tabulka 5.1: Porovnání výsledků dotazníků



Obrázek 5.1: Grafické porovnání získaného hodnocení

92 % uživatelům vyhovuje více nový formulář, než ten starý. Doba potřebná k vytvoření faktury se zkrátila ze 46 sekund na 39. Zpětnou vazbu od uživatelů lze tedy hodnotit jedině pozitivně, pohoršila si jen výdajová část, ve které ale žádné změny neproběhly.

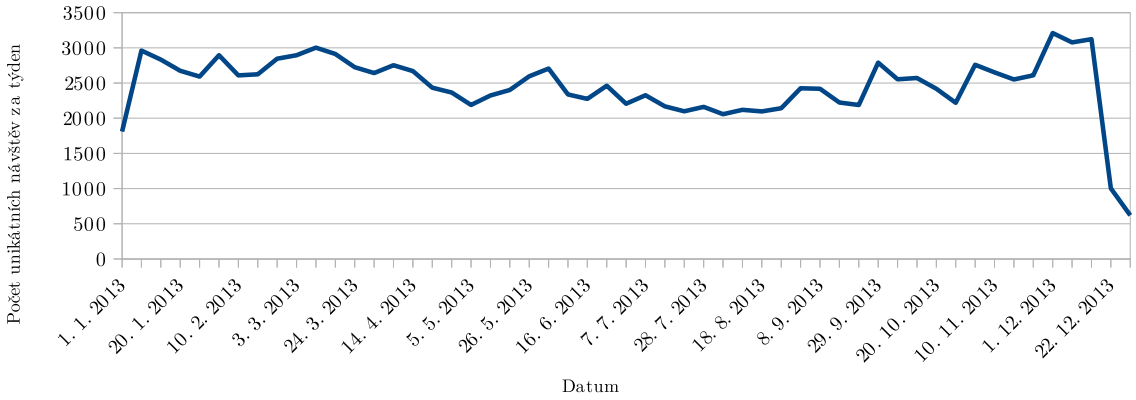
I do textové části dotazníku uživatelé psali spíše děkovné a pochvalné texty, dokonce se objevilo několik nabídek na dobrovolný finanční příspěvek.

5.2 Vliv na návštěvnost portálu

Kromě subjektivní zpětné vazby uživatelů je třeba zanalyzovat i číselné ukazatele týkající se návštěvnosti. V tomto případě budeme vycházet především z dat získaných pomocí *Google Analytics*, což je volně dostupná služba, který dokáže dokáže sesbírat data o návštěvnících a na základě nich stavit různé ukazatele. Statistiky o počtu vytvořených faktur a registracích budou brány přímo z interního logu serveru.

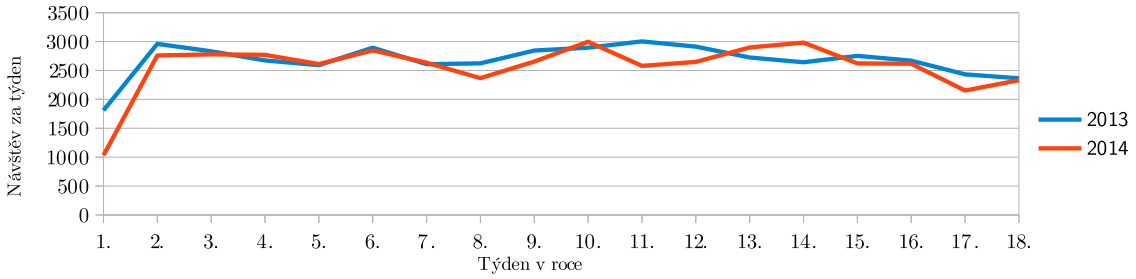
Způsob vyhodnocení

Počet návštěv se velmi mění i v průběhu týdne, protože třeba v neděli fakturuje opravdu málokdo. Díky těmto výkyvům a posunu dnů v týdnu vzhledem k datům mezi jednotlivými roky jsou grafy s denní přesností velmi špatně čitelné. Hodnoty získané za týden tento problém eliminují, přitom tím neztratíme žádnou důležitou informaci.

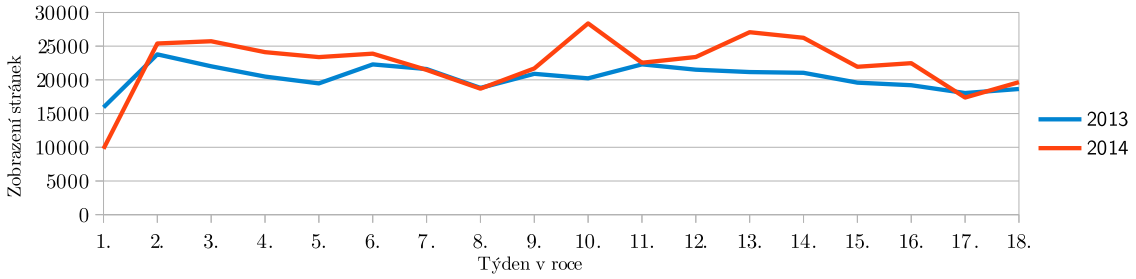


Obrázek 5.2: Počet unikátních návštěv za týden v roce 2013

Jak je vidět v grafu 5.2, návštěvnost portálu se v průběhu roku mění podle toho, jak uživatelé potřebují vytvářet faktury. Vývoj křivky je každý rok rok stejný, v létě se na ní negativně projeví dovolené, před koncem roku pozitivně Vánoce (především díky uživatelům z řad internetovým obchodům) a konec účetního roku. Porovnávání dat v průběhu roku tedy nemá žádný význam, jedeným relevantním způsobem vyhodnocením změn je porovnání návštěvnosti ve stejném období jiného roku. Bylo zvoleno 1. 1. až 30. 4. v letech 2013 a 2014. Od počátku období byl nasazen nový formulář, ostatní se změny se objevily až později a většinu času byly k dispozici jen testovací skupině. Zda budou mít i ty nějaký viditelný dopad tedy ukáže až vývoj ukazatelů v dalších obdobích.



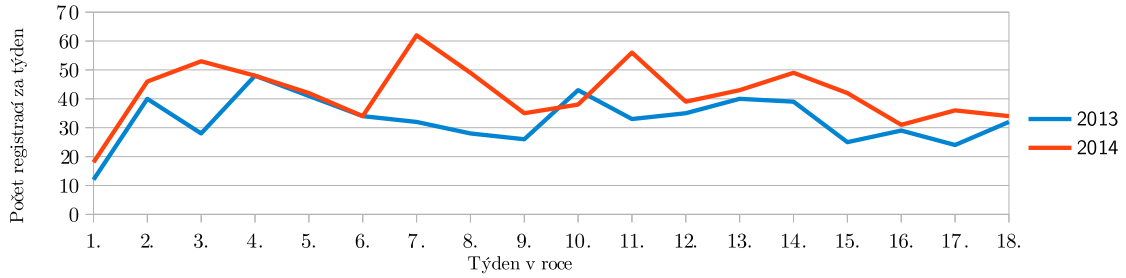
Obrázek 5.3: Porovnání počtu návštěv za týden



Obrázek 5.4: Porovnání počtu zobrazení stránek za týden

Počet unikátních návštěv ve sledovaném období klesl o 2,72 % (viz graf 5.3), zatímco počet zobrazení stránek stoupl o 11,34 % (viz graf 5.4). I když pokles návštěv vypadá na první pohled jako negativní ukazatel, sám o ničem nevypovídá. *Google Analytics* jsou k lepšímu pochopení toho ukazatele schopny sledovat dvě zajímavé a užitečné metriky. **Procento nových návštěv**, tj. kolik procent uživatelů navštívilo web poprvé a **Míru okamžitého opuštění**, která sleduje, kolik lidí web opustilo ihned bez jakékoli další interakce, jinak řečeno načetli si jen jednu jeho stránku a okamžitě se vrátili zpět.

Zatímco v roce 2013 bylo nových návštěv 36,47 %, o rok později už jen 26,54 %. Tedy byl zaznamenán značný pokles. Jenže zároveň s ním kleslo o přibližně stejnou hodnotu i procento návštěvníků okamžitě opouštějících web. Ještě v roce 2013 to bylo 32,64 %, v roce 2014 23,83 %. Z toho lze usoudit, že se spíše jen zpřesnily výsledky některého z vyhledávačů a přestal na web přivádět zákazníky, kteří o službu stejně neměli zájem a okamžitě ji opouštěli.



Obrázek 5.5: Porovnání počtu registrací

Této domněnce odpovídá také porovnání počtu registrací (viz graf 5.5), kde byl nárůst 28,18% a bude pravděpodobně důležitějším ukazatelem, než jen prostá návštěvnost webu.

Kapitola 6

Závěr

Cílem diplomové práce bylo zjistit, zda se vyplatí modernizovat zavedený webový portál a jak nejlépe tuto činnost realizovat. Prvním krokem bylo zjistit spokojenost uživatelů se současným stavem aplikace a jejich požadavky na případné změny nebo vylepšení. Průzkum dotazníkem se osvědčil, množství zpětné vazby dokonce překonalo původní očekávání a tak se získaná data dala bezpečně považovat za relevantní názor návštěvníků portálu. Mimo to bylo doplňujícími otázkami od uživatelů zjištěno, jaké změny a vylepšení by chtěli, byl sestaven seznam vylepšení, která jim byla předložena. Svým hlasováním poté určili, která z nich se bodou realizovat.

Druhým krokem bylo zvolení metody implementace. Některé uživatele by časté změny webu rozhodně nepotěšily, na druhou stranu náhlá velká změna, která by nebyla v něčem domyšlená, by je mohla zklamat ještě více. Proto byla vybrána skupinka dobrovolníků, kteří v úvodním dotazníků projevili zájem o testování novinek a v této uzavřené skupině byl použit přírůstkový model. Změny si jednotlivě prošly celým vodopádem vývoje od návrhu po testování. A až takto otestovaný a vyladěný produkt byl nasazen do ostré verze.

V průběhu detailního návrhu a implementace jednotlivých vylepšení se žádné větší problémy nevyskytly, bylo jen potřeba vyřešit velké množství těch menších. Někdy se zdánlivě jednoduchý problém značně zkomplikoval, jindy byly složitě vypadající věci rychle hotové. Zpětná vazba od testovací skupiny byla celou dobu veskrze pozitivní a tento způsob vývoje se celkově velmi osvědčil.

Analýza dopadů změn na návštěvnost webu nebyla zcela jednoznačná, ale nových uživatelů portálu pomalu ale jistě stále přibývá a to bez jakékoli formy propagace. Celkově se odvažuji hodnotit projekt modernizace pozitivně, už jen díky tomu, že se podařilo zapracovat většinu hlavních připomínek některých uživatelů bez toho, aby to mělo negativní dopad na ty ostatní. Ani tento základní úkon často velké množství firem nezvládá a potýkají se s negativními reakcemi na změny, které jsem zde vůbec nezaznamenal.

6.1 Další možnosti rozvoje

Udržet projekt konkurenceschopný nebude jednoduché, už jen tím že jeho soupeři jsou velké softwarové firmy jako například *CÍGLER SOFTWARE, a.s.*, které se výrobou účetního software zabývají desítky let, mají v tomto oboru mnohem větší praxi a disponují nesrovnatelně většími zdroji.

Možnou variantou je dokončení zatím nerealizovaných návrhů na úpravy systému. Poněkud riskantním krokem by mohl být redesign celého portálu s důrazem na jednoduchost.

Současná podoba obsahuje velké množství textů, které mu sice zajišťují přední pozice ve vyhledávačích, ale na uživatele mohou působit složitě a děsivě. Jedním z možných, ale náročných rozšíření, by byl mobilní web, který by uživatelům umožnil pohodlně vytvářet faktury i z chytrých telefonu a tabletů.

Jak ukázaly analýzy, v tomto roce na web přichází méně nových návštěvníků, než v předchozích letech, proto by stálo za to zvážit nějaké další formy propagace, které by na web přitáhly více uživatelů. Logicky by se jevila i expanze do zahraničí, která je ale v tomto případě značně komplikována nutností nastudovat si místní legislativu. Rozsah této komplikace jsem podcenil už v počátku tohoto projektu a v jiné zemi by pravděpodobně problémů nebylo o moc méně.

Ať bude zvolena kterákoli z variant, prvním a správným krokem zcela určitě byla modernizace portálu tak, aby odpovídal současným standardům a požadavkům uživatelů.

Literatura

[1] Adobe Systems Incorporated: Document management – Portable document format. http://wwwimages.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/|PDF32000_2008.pdf, [cit. 2013-12-22].

[2] Clarke, J.: *SQL Injection Attack and defense*. Syngress Publishing, Inc, 2009, iSBN 978-1-59749-424-3.

[3] Crockford, D.: The application/json Media Type for JavaScript Object Notation (JSON). [cit. 2014-04-24].

[4] Dahl, D.: Privacy/Features/DOMCryptAPISpec/Latest. https://wiki.mozilla.org/Privacy/Features/DOMCryptAPISpec/Latest, [cit. 2014-04-28].

[5] Demmel, J.: Basic Issues in Floating Point Arithmetic and Error Analysis. http://www.cs.berkeley.edu/ demmel/cs267/lecture21/lecture21.html, [cit. 2014-02-05].

[6] Gilmore, W. J.: *Velká kniha PHP 5 a MySQL*. Zoner Press, 2005, iSBN 80-86815-20-X.

[7] Hanáček, P.: BIS 1. https://www.fit.vutbr.cz/study/courses/BIS/private/bis01.pdf, [cit. 2014-05-03].

[8] Lambert, J.: Introducing CAPICOM. http://msdn.microsoft.com/en-us/library/ms995332.aspx, [cit. 2014-04-28].

[9] Microsoft: Alternatives to Using CAPICOM. http://msdn.microsoft.com/en-us/library/cc778518.aspx, [cit. 2014-04-28].

[10] Ministerstvo financí ČR: XML služby ARES. http://wwwinfo.mfcr.cz/ares/ares_xml.html.cz, [cit. 2013-11-23].

[11] N. Freed: Multipurpose Internet Mail Extensions. http://tools.ietf.org/html/rfc2045, [cit. 2013-12-28].

[12] Oracle: What Applets Can and Cannot Do. http://docs.oracle.com/javase/tutorial/deployment/applet/security.html, [cit. 2013-12-22].

[13] Oracle Corporation: The History of Java Technology. [cit. 2014-03-24].

[14] Oracle Corporation: 2 Understanding Just-In-Time Compilation and Optimization. [cit. 2014-03-27].

[15] Shafranovich, Y.: Common Format and MIME Type for Comma-Separated Values (CSV) Files. http://tools.ietf.org/html/rfc4180, October 2005.

[16] Správa základních registrů : Registr osob. http://www.szrcr.cz/registr-osob, [cit. 2014-05-05].

[17] The PHP Group: Output Control Functions. http://www.php.net/manual/en/ref.outcontrol.php, [cit. 2013-12-04].

[18] The PHP Group: PHP: Database issues - Manual. http://cz2.php.net/manual/en/faq.databases.php, [cit. 2013-12-28].

[19] The PHP Group: htmlspecialchars. http://www.php.net/manual/en/function.htmlspecialchars.php, [cit. 2014-05-09].

[20] W3C: JavaScript Web APIs. http://www.w3.org/standards/webdesign/script, [cit. 2013-12-28].

[21] W3C: Extensible Markup Language (XML) 1.0 (Fifth Edition). http://www.w3.org/TR/2008/REC-xml-20081126/, [cit. 2014-04-24].

[22] W3C: Web Cryptography Working Group. http://www.w3.org/2012/webcrypto/, [cit. 2014-04-28].

[23] WWW stránky: rsync. http://rsync.samba.org/, [cit. 2013-12-01].

[24] WWW stránky: QR Platba. http://qr-platba.cz/, [cit. 2013-12-05].

[25] WWW Stránky: Privacy/Features/DOMCryptAPISpec/Latest. http://phpqrcode.sourceforge.net/, [cit. 2014-04-29].

[26] WWW stránky: QRcode.com. [cit. 2014-05-08].

Příloha A

Obsah CD

Na vloženém CD jsou tyto soubory a adresáře:

- `/src/` – zdrojové kódy
  - `/src/README` – soubor s návodem na instalaci
  - `/src/www/` – zdrojové kódy portálu
  - `/src/dtb.sql` – soubor pro vytvoření databáze
  - `/src/java/` – zdrojové kódy Java appletu
  - `/src/config/` – konfigurační soubory pro instalaci
- `/poster/` – plakát v *PDF* a *SVG*
- `/thesis/` – zdrojové kódy diplomové práce
- `/thesis.pdf` – diplomová práce v *PDF*